PENGEMBANGAN APLIKASI PENENTUAN PRIORITAS KEBUTUHAN FUNGSIONAL PERANGKAT LUNAK BERDASARKAN KEBUTUHAN NON-FUNGSIONAL

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Dini Indah Nurul Rizki Pancawati Raharjo
NIM: 145150200111074

PROGRAM STUDI TEKNIK NFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENGEMBANGAN APLIKASI PENENTUAN PRIORITAS KEBUTUHAN FUNGSIONAL PERANGKAT LUNAK BERDASARKAN KEBUTUHAN NON-FUNGSIONAL

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Dini Indah Nurul Rizki Pancawati Raharjo
NIM: 145150200111074

Skripsi ini telah diuji dan dinyatakan lulus pada
28 Desember 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Tri Astoto Kurniawan, S.T, M.T, Ph.D	Denny Sagita Rusdianto, S.Kom., M.Kom
NIP: 19710518 200312 1 001	NIP: 19851124 201504 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 28 Desember 2018

Dini Indah Nurul Rizki Pancawati Raharjo

NIM: 145150200111074

PRAKATA

Puji syukur penulis panjatkan atas kehadirat Allah SWT atas limpahan rahmat serta karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "PENGEMBANGAN APLIKASI PENENTUAN PRIORITAS KEBUTUHAN FUNGSIONAL PERANGKAT LUNAK BERDASARKAN KEBUTUHAN NON-FUNGSIONAL". Melalui pengantar ini, penulis hendak menyampaikan terima kasih kepada:

1. Allah SWT yang telah memberikan nikmat waktu, kesempatan, serta hidayah ilmu dan atas kuasa-Nya penulis bisa menyelesaikan skripsi ini.
2. Orang tua penulis atas doa, kasih sayang, serta dukungan yang selalu diberikan pada penulis dengan tulus dan tanpa henti.
3. Bapak Tri Astoto Kurniawan, S.T.,M.T.,Ph.D, selaku Ketua Jurusan Teknik Informatika sekaligus Dosen Pembimbing I yang telah meluangkan waktu untuk membimbing, mengarahkan, dan memberi nasihat kepada penulis sehingga penulis dapat menyelesaikan skripsi ini.
4. Bapak Denny Sagita R., S.Kom, M.Kom, selaku Dosen Pembimbing II, yang telah meuangkan waktu dan dengan sabar mengarahkan, membina, dan membimbing penulis sehingga dapat menyelesaikan skripsi ini.
5. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D, selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
6. Bapak Agus Wahyu Widodo, S.T, M.Cs, selaku Ketua Program Studi Teknik Informatika Universitas Brawijaya.
7. Keluarga penulis yang senantiasa memberikan dorongan, kasih sayang, serta doa yang tulus sehingga penulis dapat menyelesaikan skripsi ini.
8. Rekan-rekan penulis, yaitu Nauli Cahyaning Mekarsari, Alifa Nurlianti, Kristiana Meida Lestari, Derinda Pebriyanti, Jessy Ratna Wulandari, Nurhaida Syadrina yang senantiasa menyemangati, menemani, dan menghadirkan hangatnya persahabatan.
9. Rekan-rekan teknik informatika 2014 dan POROS yang senantiasa membantu penulis dan mewarnai hari-hari di FILKOM.

Penulis menyadarai bahwa skripsi ini masih jauh dari kata sempurna. Oleh karena itu, penulis mengharapkan kritik serta saran yang membangun dari pembaca agar skripsi ini bisa lebih baik lagi.

Malang, 28 Desember 2018

Dini Indah Nurul Rizki Pancawati Raharjo

diniindah03@gmail.com

ABSTRAK

Dini Indah Nurul Rizki Pancawati Raharjo, Pengembangan Aplikasi Penentuan Prioritas Kebutuhan Fungsional Perangkat Lunak Berdasarkan Kebutuhan Non-Fungsional

Pembimbing: Tri Astoto Kurniawan, S.T., M.T., Ph.D, Denny Sagita Rusdianto, S.Kom., M.Kom.

Kebutuhan perangkat lunak dapat didefinisikan sebagai layanan yang harus disediakan perangkat lunak untuk mendukung tercapainya tujuan pengguna dalam menggunakan perangkat lunak tersebut. Kebutuhan perangkat lunak merupakan dasar dalam proses pengembangan perangkat lunak. Penentuan prioritas kebutuhan merupakan proses yang penting untuk dilakukan terutama dalam situasi dimana waktu dan sumber daya pengembangan perangkat lunak terbatas. Penentuan prioritas ini bermanfaat untuk menyediakan rekomendasi urutan implementasi kebutuhan dan juga kebutuhan yang harus mendapatkan perhatian lebih dalam pengimplementasiannya. Penelitian yang dilakukan oleh Umang Garg dan Abhishek Singhal mengusulkan suatu metode penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional. Penelitian ini menerapkan metode yang diusulkan Garg dan Singhal dalam bentuk aplikasi berbasis web, dengan tujuan untuk membantu proses penentuan prioritas kebutuhan. Penentuan prioritas kebutuhan dimulai dengan melakukan *pair-wise comparison* antara kebutuhan non-fungsional, kemudian dilakukan *pair-wise comparison* antara kebutuhan fungsional dan kebutuhan non-fungsional, terakhir didapatkan urutan kebutuhan berdasarkan prioritasnya serta nilai prioritasnya. Pengujian sistem dilakukan dengan pengujian unit terhadap tiga *method* pada sistem, pengujian integrasi, serta pengujian validasi yang terdiri dari 27 kasus uji yang menghasilkan hasil valid.

Kata kunci: penentuan prioritas kebutuhan, kebutuhan fungsional, kebutuhan non-fungsional

ABSTRACT

Dini Indah Nurul Rizki Pancawati Raharjo, Development of Software Functional Requirements Prioritization Based on Non-Functional Requirements

Supervisors: Tri Astoto Kurniawan, S.T., M.T., Ph.D, Denny Sagita Rusdianto, S.Kom., M.Kom.

Software requirements can be defined as services that software must provide in order to support user achieving the goal of using the software. Software requirements are the basis in the software development process. In situation where the source and time are limited it is important to prioritize the requirements. Requirements prioritization shows which requirement should be implemented first and get more attention in the implementation process. A research by Umang Garg and Abhishek Singhal, a method to prioritize software requirements based on non-functional requirements is proposed. In this research, an application to prioritize software functional requirements is developed in order to help the process of requirements prioritization, the method used in this research is the one that proposed by Umang Garg and Abhishek Singhal. Requirement prioritization started with a pair-wise comparison between the non-functional requirements, and then a pairwise comparison between functional requirements with non-functional requirements, lastly the order of requirements' priority and the value of priority are obtained. The system is tested by unit testing on 3 methods of the system, integration testing, and 27 test cases on validation testing which give valid results.

Keywords: *requirement prioritization, functional requirement, non-functional requirement*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Perangkat lunak merupakan kumpulan instruksi dalam suatu bahasa pemrograman yang menjembatani pengguna suatu perangkat keras dengan perangkatnya dan memungkinkan perangkat keras untuk melakukan fungsi-fungsi pemrosesan berdasarkan data yang dimasukkan pengguna serta memberikan hasil yang diharapkan oleh pengguna. Perangkat lunak dibangun melalui proses rekayasa perangkat lunak, yaitu disiplin rekayasa yang berfokus pada semua aspek produksi perangkat lunak yang di dalamnya terdiri dari proses spesifikasi kebutuhan perangkat lunak, perancangan dan implementasi perangkat lunak, validasi perangkat lunak, dan evolusi perangkat lunak (Sommerville, 2011).

Tahapan spesifikasi kebutuhan dalam rekayasa perangkat lunak menghasilkan daftar kebutuhan yang mendeskripsikan kebutuhan pengguna serta pemangku kepentingan dan menjadi dasar untuk tahapan rekayasa selanjutnya. Kebutuhan perangkat lunak merupakan bagian paling utama dan krusial dalam proses rekayasa serta menentukan tingkat kesuksesan penerimaan perangkat lunak oleh klien (Longani, et al., 2017). Terdapat dua jenis kebutuhan sistem dalam pengembangan perangkat lunak, yaitu kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan fungsional mendeskripsikan aktivitas-aktivitas yang harus dapat dilakukan sistem (Satzinger, et al., 2012), sedangkan kebutuhan non-fungsional merupakan kebutuhan yang tidak secara langsung berhubungan dengan kemampuan sistem namun berhubungan dengan properti-properti dari perangkat lunak (misalnya *response time*, *security*) (Sommerville, 2011). Daftar kebutuhan perangkat lunak didapatkan dengan memahami kebutuhan dari sudut pandang, peran, dan objektif pemangku kepentingan (Pandey, et al., 2010).

Penentuan prioritas kebutuhan menjadi hal yang penting dilakukan agar pengembangan perangkat lunak dapat berlangsung dengan lebih efisien dan mengurangi konflik yang muncul di antara pemangku kepentingan, karena prioritas dan urgensi dari setiap kebutuhan dapat disetujui oleh semua pihak (Liaqat, et al., 2016). Penentuan prioritas kebutuhan juga memegang peranan penting dalam perencanaan rilis perangkat lunak, mengkombinasikan strategi untuk anggaran biaya dan penjadwalan, juga strategi pemasaran (Perini, et al., 2013).

Salah satu permasalahan yang bisa terjadi jika tidak dilakukan penentuan prioritas kebutuhan adalah *scope creep*, yaitu kondisi dimana kebutuhan terus diajukan oleh klien sehingga *scope* semakin melebar (Satzinger, et al., 2012). *Paper* Liaqat, at al., (Liaqat, et al., 2016) memapaparkan satu contoh kegagalan proyek pengembangan perangkat lunak yang disebabkan oleh kebutuhanyang terus bertambah dan *scope* yang terus melebar, yaitu pada proyek FBI *Virtual Case File* pada tahun 2005 yang mengakibatkan kerugian kurang lebih 170 miliar dollar dan akhirnya dihentikan setelah lima tahun pengembangan.

Penentuan prioritas kebutuhan menjadi konsep yang populer dalam rekayasa perangkat lunak dalam 30 tahun terakhir. Metode penentuan prioritas kebutuhan banyak diajukan untuk meningkatkan implementasi konsep ini pada organisasi (Herrmann & Daneva, 2008). Konsep-konsep populer dalam penentuan prioritas kebutuhan di antaranya adalah AHP (*Analyctical Hierarchy Process*), *Numerical Assignment*, HCV (*Hierarchy Cummulative Voting*), MosCoW *Technique*, dan *Bubble Sort Technique*. Kekurangan dari metode AHP adalah prosesnya memakan waktu yang lama, sedangkan metode *numerical assignment* memberikan nilai prioritas dalam bentuk kelompok, sehingga tidak ada prioritas kebutuhan secara individu.

Penelitian yang dilakukan oleh Umang Garg dan Abhishek Singhal (Garg & Singhal, 2017) mengusulkan konsep penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan pada kebutuhan non-fungsional perangkat lunak. Dalam penelitian ini, Garg dan Singhal menggunakan tiga metode yang sudah ada sebelumnya, yaitu *Analyctical Hierarchy Process* (AHP)a, *Cost-Value Approach*, dan *Numerical Assignments* (*Grouping*) untuk memanfaatkan kelebihan yang ditawarkan setiap teknik tersebut. Penelitian ini dilakukan agar pengembang perangkat lunak dapat mengambil kesimpulan mengenai kebutuhan mana yang lebih baik diimplementasikan terlebih dahulu dengan memperhatikan juga tingkat kepentingan dari kebutuhan non-fungsional yang ada.

Penelitian ini mengambil judul "Pengembangan Aplikasi Penentuan Prioritas Kebutuhan Fungsional Perangkat Lunak Berdasarkan Kebutuhan Non-Fungsional" dengan meneraplan metode dari penelitian Garg dan Shinghal yang berjudul "*Software Requirement Prioritization Based on Non-Functional Requirements*" ke dalam bentuk aplikasi *web*. Harapannya adalah hasil dari penelitian ini dapat membantu proses penentuan prioritas kebutuhan agar lebih cepat dan mudah pada pengembangan perangkat lunak serta lebih memperhatikan kebutuhan non-fungsional perangkat lunak.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dipaparkan di atas, dapat dirumuskan masalah sebagai berikut:

1. Bagaimana hasil rekayasa kebutuhan dalam pengembangan aplikasi penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional?
2. Bagaimana hasil perancangan dan implementasi aplikasi penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional?
3. Bagaimana hasil pengujian menjawab kebutuhan yang diperlukan untuk pengembangan aplikasi penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional?

1.3 Tujuan

Tujuan dilaksanakannya penelitian ini adalah sebagai berikut:

1. Mengembangkan sebuah aplikasi yang diharapkan dapat membantu analis atau tim pengembang perangkat lunak dalam menentukan prioritas kebutuhan fungsional perangkat lunak.

1.4 Manfaat

Manfaat dari penelitian ini adalah:

1. Menyediakan aplikasi yang dapat membantu dalam menentukan prioritas kebutuhan perangkat lunak yang memperhatikan kebutuhan non-fungsional.

1.5 Batasan masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Pengembangan aplikasi dalam penelitian ini dilakukan berdasarkan *paper* yang ditulis oleh Umang Garg dan Abhishek Singhal yang berjudul "*Software Requirement Prioritization based on Non-Functional Requirements*" (Garg & Singhal, 2017).
2. Model pengembangan yang digunakan adalah model *waterfall*. Pengembangan dilakukan hingga tahap pengujian.
3. Aplikasi yang dikembangkan menggunakan *platform* web.
4. *Framework* yang digunakan dalam pengembangan aplikasi adalah CodeIgniter dan MaterializeCSS.

1.6 Sistematika pembahasan

Untuk mempermudah memahami laporan ini, penulis membuat sistematika pembahsan yang menjelaskan secara singkat mengenai isi dari setiap bab dalam laporan ini:

BAB 1: PENDAHULUAN

Bab ini menjelaskan latar belakang pengembangan aplikasi penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika pembahasan dalam laporan.

BAB 2: LANDASAN KEPUSTAKAAN

Bab ini menjelaskan kajian pustaka dari penelitian-penelitian terkait serta dasar teori yang digunakan sebagai referensi dalam penelitian ini.

BAB 3: METODOLOGI PENELITIAN

Bab ini membahas mengenai langkah-langkah penelitian dalam pengembangan aplikasi penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional.

BAB 4: REKAYASA KEBUTUHAN

Bab ini membahas proses dan hasil analisis kebutuhan yang diperlukan untuk mengembangkan aplikasi penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional. Bab ini juga menampilkan pemodelan kebutuhan dalam bentuk *use case diagram* dan *use case scenario*.

BAB 5: PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskan mengenai perancangan dan hasil implementasi aplikasi. Perancangan terdiri dari perancangan arsitektur, perancangan komponen, perancangan data, dan perancangan antarmuka. Sedangkan implementasi terdiri dari implementasi kode program, implementasi data, serta implementasi antarmuka.

BAB 6: PENGUJIAN

Bab ini menguraikan pengujian dan hasil pengujian yang dilakukan pada aplikasi. Pengujian yang dilakukan berupa pengujian unit, integrasi, dan validasi.

BAB 7: PENUTUP

Bab ini menjelaskan kesimpulan dari hasil penelitian yang sudah dilakukan serta saran untuk pengembangan selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan terdiri dari kajian pustaka mengenai penelitian-penelitian yang sudah dilakukan sebelumnya dan berhubungan dengan penelitian yang dilakukan, serta teori-teori yang menjadi referensi dalam pengerjaan penelitian ini.

2.1 Kajian Pustaka

Penentuan prioritas kebutuhan menjadi konsep yang populer dalam 30 tahun terakhir, banyak metode yang diajukan untuk meningkatkan pengimplementasian konsep ini pada organisasi (Herrmann & Daneva, 2008). Penentuan prioritas kebutuhan dianggap penting karena sumber daya dan waktu dalam pengembangan perangkat lunak terbatas. Penentuan prioritas ini dapat memastikan bahwa kebutuhan yang lebih utama dan lebih tinggi prioritasnya diimplementasikan lebih dulu.

Penelitian yang berjudul "*A Machine Learning Approach to Software Requirements Prioritization*" (Perini, et al., 2013) memaparkan metode penentuan priotitas kebutuhan yang dilakukan dengan menggunakan suatu metode yang disebut CBRank (*Case-Based Ranking*). Metode CBRank ini mengkombinasikan preferensi dari pemangku kepentingan dengan perkiraan susunan kebutuhan yang dihitung dengan menggunakan teknik *machine learning*. Melalui metode ini, peran manusia dalam memasukkan informasi berupa preferensi dapat dikurangi sehingga meningkatkan keakuratan dalam estimasi peringkat final, dan pengetahuan domain dikodekan menjadi urutan hubungan parsial yang didefinisikan berdasarkan atribut kebutuhan dapat dieksploitasi sehingga mendukung proses elisitasi yang adaptif.

Penelitian yang berjudul "*Requirement Prioritization using Adaptive Fuzzy Hierarchical Cumulative Voting*" (Jawale, et al., 2017) memaparkan metode penentuan priotitas dengan *Adaptive Fuzzy Hierarchial Cumulative Voting* (AFHCV) yang menggunakan mekanisme adaptif dengan *Fuzzy Hierarchial Cumulative Voting* (FHCV) untuk meningkatkan cakupan *events* yang terjadi saat *runtime*. Mekanisme adaptif meliputi penambahan set kebutuhan baru, analisis dan realokasi kebutuhan, penempatan dan perubahan prioritas dan penentuan priotitas ulang. Penentuan prioritas ulang bertujuan untuk meningkatkan hasil dari teknik AFHCV yang diusulkan. Berdasarkan perbandingan yang dilakukan antara AFHCV dan FHCV, didapatkan bahwa teknik AFHCV memberikan hasil yang lebih baik dari FHCV.

Penentuan prioritas kebutuhan dalam penelitian-penelitian terkait yang sudah disebutkan di atas kurang memperhatikan kebutuhan non-fungsional perangkat lunak dalam prosesnya. Penelitian ini menerapkan metode yang terdapat dalam penelitian yang dilakukan oleh Umang Garg dan Abhishek Singhal (2017) yang berjudul "*Software Requirements Prioritization Based on Non-Functional Requirements*". Metode yang diajukan Garg dan Shinghal terdiri dari tiga langkah

utama, yaitu memberi nilai kepentingan dan melakukan *pair-wise comparison* pada kebutuhan non-fungsional dengan menggunakan skala *pair-wise comparison* dalam metode AHP, kemudian memberi nilai kepentingan pada kebutuhan fungsional dengan memperhatikan kebutuhan non-fungsional dengan menggunakan skala *Numrical Assignment* (*Grouping*), dan terakhir menghitung nilai prioritas dengan perkalian matriks.

2.2 Kebutuhan Perangkat Lunak

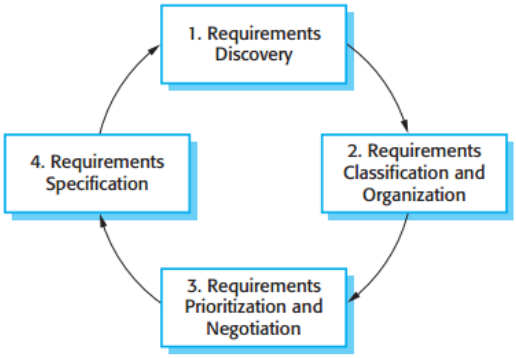
Tahapan yang pertama kali dilakukan dalam pengembangan perangkat lunak adalah menganalisis dan mendefinisikan kebutuhan perangkat lunak. Kebutuhan perangkat lunak merupakan deskripsi mengenai hal-hal yang harus dapat dilakukan oleh perangkat lunak, baik layanan maupun batasan- batasan dari operasi di dalamnya (Sommerville, 2011). Berdasarkan levelnya, terdapat dua jenis kebutuhan, yaitu kebutuhan pengguna (*user requirements*) dan kebutuhan sistem (*system requirements*). Kebutuhan pengguna merupakan pernyataan dalam bahasa natural yang menyatakan layanan dan batasan- batasan yang diharapkan pengguna disediakan oleh sistem. Sedangkan kebutuhan sistem merupakan deskripsi fungsi-fungsi sistem, layanan-layanan sistem, dan juga batasan-batasan sistem yang disusun dalam suatu dokumen spesifikasi kebutuhan perangkat lunak, yang kemudian harus diimplementasikan dalam sistem (Sommerville, 2011). Kebutuhan sistem meliputi seluruh fungsi yang harus disediakan sistem (kebutuhan fungsional) dan juga hal-hal yang terkait format antarmuka pengguna dan kebutuhan mengenai *reliability*, performa sistem, dan juga keamanan sistem (kebutuhan non-fungsional) (Satzinger, et al., 2012).

2.2.1 Proses Rekayasa Kebutuhan

Proses untuk menemukan, menganalisa, mendokumentasikan, dan juga memeriksa kebutuhan-kebutuhan perangkat lunak disebut rekayasa kebutuhan (Sommerville, 2011). Proses rekayasa kebutuhan meliputi empat tahap, yaitu penilaian apakah sistem berguna bagi bisnis (studi kelayakan), penggalian kebutuhan (elisitasi dan analisis), membuat kebutuhan ke dalam bentuk yang standar (spesifikasi), dan memastikan semua kebutuhan mendefinisikan sistem yang diinginkan klien (validasi) (Sommerville, 2011).

2.2.1.1 Elisitasi dan Analisis Kebutuhan

Tahap elisitasi dan analisis kebutuhan merupakan tahap dimana perekayasa perangkat lunak bekerja dengan klien untuk mengetahui domain aplikasi, layanan yang harus disediakan sistem, performa yang diperlukan, batasan perangkat keras, dan lain sebagainya (Sommerville, 2011). Pemangku kepentingan banyak dilibatkan dalam tahap ini.



Gambar 2.1 Proses Elisitasi dan Analisis Kebutuhan

Sumber: (Sommerville, 2011)

Dalam proses elisitasi dan analisis kebutuhan ini terdapat empat tahap utama seperi terlihat pada Gambar 2.1, penjelasan dari setiap tahapannya adalah sebagai berikut:

a. *Requirement Discovery*

Tahap *requirement discovery* disebut juga elisitasi kebutuhan. *Requirement discovery* merupakan proses untuk mengumpulkan informasi mengenai sistem yang diperlukan dan juga sistem yang sudah ada, juga menyaring pengguna dan kebutuhan sistem (Sommerville, 2011). Sumber-sumber informasi dalam kegiatan ini bisa didapat dari dokumentasi, pemangku kepentingan sistem, dan spesifikasi dari sistem yang mirip.

b. *Requirements Classification and Organization*

Proses yang dilakukan pada tahap ini adalah mengelompokkan kebutuhan yang saling berhubungan ke dalam kelompok yang koheren (Sommerville, 2011). Cara yang paling sering digunakan dalam pengelompokkan kebutuhan ini adalah dengan menggunakan model arsitektur sistem untuk mengidentifikasi subsistem-subsistem dan mengasosiasikan kebutuhan dengan subsistem yang sesuai.

c. *Requirement Prioritization and Negotiation*

Proses penggalian kebutuhan dari klien ataupun pemangku kepentingan lainnya menghasilkan banyak daftar kebutuhan yang jumlahnya proporsional dengan jumlah fitur yang harus diimplementasikan oleh tim pengembang perangkat lunak dalam kurun waktu dan sumber daya yang tersedia (Liaqat, et al., 2016). Kebutuhan yang didapat dari banyak pemangku kepentingan ini bisa jadi menghasilkan konflik antar kebutuhan. Tahap *requirement prioritization and negotiation* berisi kegiatan negosiasi untuk memprioritaskan kebutuhan dan menemukan juga menyelesaikan konflik yang ada (Sommerville, 2011). Para pemangku kepentingan bertemu untuk meluruskan perbedaan dan menyetujui kebutuhan-kebutuhan yang akan diimplementasikan. Penentuan prioritas kebutuhan memegang peranan penting dalam pengembangan perangkat lunak dan juga dalam perencanaan rilis perangkat lunak, mengkombinasikan strategi

untuk anggaran biaya dan penjadwalan, juga strategi pemasaran (Perini, et al., 2013).

d. *Requirements Specification*

Tahap ini merupakan tahap dimana kebutuhan didokumentasikan ke dalam bentuk dokumen tertulis, model grafik, model formal, kumpulan skenario-skenario penggunaan perangkat lunak, *prototype*, atau kombinasi dari bentuk-bentuk ini (Pressman & Maxim, 2013). Dokumen yang dihasilkan dari tahap ini digunakan dalam tahap selanjutnya pada proses rekayasa kebutuhan.

2.2.1.2 Validasi Kebutuhan

Validasi kebutuhan merupakan tahap untuk memastikan bahwa daftar kebutuhan benar-benar mendefinisikan sistem yang akan dibangun (Sommerville, 2011). Sepesifikasi sistem dan informasi lain yang didapat dari proses rekayasa kebutuhan lainnya dinilai kualitasnya dalam tahap ini. Validasi kebutuhan menguji spesifikasi yang sudah didapat untuk memastikan bahwa semua kebutuhan sistem tidak ambigu; memastikan semua inkonsistensi, ketidaklengkapan, dan eror sudah dideteksi dan dikoreksi; dan juga memastikan semua produk kerja telah sesuai dengan standar yang ditetapkan untuk proses, proyek, dan produk (Pressman, 2001). Pihak-pihak yang melakukan validasi ini adalah tim perekayasa sistem, klien, pengguna, dan juga pemangku kepentingan lainnya. Validasi ini dapat dilakukan dengan menggunakan *checklist* dari kondisi-kondisi yang diinginkan.

2.2.2 Metode Penentuan Prioritas Kebutuhan Fungsional Perangkat Lunak Berdasarkan Kebutuhan Non-Fungsional

Umang Garg dan Abhishek Singhal (2017) dalam penelitiannya yang berjudul "*Software Requirements Prioritization Based on Non-Functional Requirements*" mengajukan metode penentuan prioritas kebutuhan perangkat lunak dengan memperhatikan kebutuhan non-fungsional yang ada. Kebutuhan perangkat lunak yang diproritaskan pada *paper* tersebut adalah kebutuhan fungsional. Langkah-langkah dalam metode yang diajukan Garg dan Singhal, yaitu:

1. Melakukan *pair-wise comparison* antara kebutuhan non-fungsional dengan menggunakan skala *pair-wise comparison* dalam metode AHP. Setelah dilakukan *pair-wise comparison*, nilai prioritas kebutuhan non-fungsional didapatkan melalui normalisasi, yaitu dengan melakukan pembagian antara jumlah dari setiap nilai *pair-wise comparison* pada tiap elemen (satu kolom) dengan nilai *pair-wise comparison* pada elemen individu (nilai setiap baris). Persamaan untuk menghitung nilai prioritas kebutuhan non-fungsional terdapat pada Persamaan 2.1

$$\boldsymbol{Prioritas\ NF[x]} = \sum_{i=1}^{n} \left(\frac{nf_{[x],i}}{k_i} \right) \qquad \textbf{(2.1)}$$

Keterangan :

NF[x] = Kebutuhan Non-fungsional [x]

$nf_{[x],i}$ = Nilai perbandingan kebutuhan non-fungsional [x] dengan kebutuhan non-fungsional i

k_i = Total nilai perbandingan pada setiap kolom

Tabel 2.1 Skala *pair-wise comparison* dalam AHP

Intensitas Kepentingan	Definisi	Penjelasan
1	Kepentingan dari kedua elemen bernilai sama.	Kedua elemen memiliki pengaruh yang sama terhadap tujuan.
3	Salah satu elemen sedikit lebih penting dari elemen lainnya.	Pengalaman juga penilaian sedikit mendukung salah satu elemen.
5	Salah satu elemen lebih penting dari elemen lainnya.	Pengalaman juga penilian sangat mendukung salah satu elemen.
7	Salah satu elemen sangat lebih penting dari elemen lainnya.	Salah satu elemen sangat didukung dari elemen lainnya dan lebih dominan dalam praktik.
9	Salah satu elemen mutlak lebih penting dari elemen lainnya.	Bukti mendukung salah satu elemen lebih diutamakan dari yang lain dan memiliki penegasan urutan yang lebih tinggi.
2,4,6,8	Nilai-nilai antara dua intensitas kepentingan yang berdekatan.	Kondisi dimana kompromi diperlukan.

Sumber: (Karlsson & Ryan, 1997)

2. Melakukan *pair-wise comparison* antara kebutuhan fungsional dengan kebutuhan non-fungsional. Skala yang digunakan untuk *pair-wise comparison* ini adalah skala yang terdapat pada metode *Numerical Assignements* (*Grouping*) pada Tabel 2.2 berikut ini:

Tabel 2.2 Skala nilai kepentingan kebutuhan fungsional berdasarkan kebutuhan non-fungsional

Nilai Kepentingan	Deskripsi
1	Kepentingan bernilai sama.

2	Nilai kepentingan sedikit lebih tinggi.
3	Nilai kepentingan lebih tinggi.
4	Nilai kepentingan sangat tinggi.
5	Nilai kepentingan mutlak lebih tinggi.

Sumber: (Garg & Singhal, 2017)

3. Menghitung prioritas dengan perkalian matriks. Perkalian matriks dilakukan antara nilai perbandingan kebutuhan fungsional yang sudah ditentukan pada langkah 2, dengan nilai prioritas dari setiap elemen kebutuhan non-fungsional. Persamaan untuk menghitung nilai prioritas kebutuhan non-fungsional terdapat pada Persamaan 2.2

$$\boldsymbol{Prioritas\ F[x]} = \sum_{i=1}^{n} \left(\boldsymbol{f_{[x],i}} \times \boldsymbol{Prioritas\ nf_i} \right) \quad \textbf{(2.2)}$$

Keterangan :

F[x] = Kebutuhan Fungsional [x]

$f_{[x],i}$ = Nilai perbandingan kebutuhan fungsional [x] dengan kebutuhan non-fungsional i

$Prioritas\ nf_i$ = Nilai prioritas kebutuhan non-fungsional i

Tujuan dari penelitian yang dilakukan Garg dan Singhal ini adalah agar pengembang perangkat lunak dapat menentukan kebutuhan mana yang lebih baik diimplementasikan terlebih dahulu dengan memberi perhatian lebih pada kebutuhan non-fungsional, seperti keamanan dan lain sebagainya (Garg & Singhal, 2017).

2.3 Pengembangan Perangkat Lunak

Pengembangan perangkat lunak (*software development*) merupakan salah satu aktivitas dalam rekayasa perangkat lunak dimana perangkat lunak dianalisa, dirancang, deprogram, dan diuji. Pengembangan perangkat lunak memiliki beberapa model yang dijadikan *framework* atau kerangka kerja, di antaranya adalah model *waterfall*, model *incremental*, model *prototype*, model *Rapid Application Development* (RAD), dan model-model lainnya. Model pengembangan perangkat lunak mendefinisikan alur dari keseluruhan aktivitas, tugas-tugas, juga aksi yang diperlukan, jumlah iterasi, produk kerja, juga susunan pekerjaan yang harus dilakukan (Pressman & Maxim, 2013). Pengembangan perangkat lunak terdiri dari enam langkah utama, yaitu pengumpulan kebutuhan dan analisis, perancangan, implementasi, pengujian, *deployment*, dan pemeliharaan (Elysium Academy Private Limited, 2017). Pengembangan perangkat lunak pun memiliki beberapa pendekatan, di antaranya adalah pendekatan terstruktur dan pendekatan berorientasi objek.

2.3.1 Model Pengembangan Perangkat Lunak

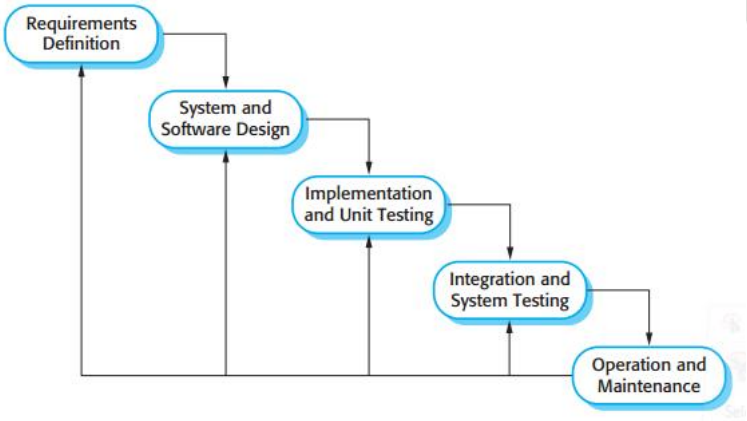
Model pengembangan perangkat lunak mendefinisikan alur dari keseluruhan aktivitas, tugas-tugas, juga aksi yang diperlukan, jumlah iterasi, produk kerja, juga susunan pekerjaan yang harus dilakukan (Pressman & Maxim, 2013). Model pengembangan perangkat lunak digunakan agar proses pengembangan perangkat lunak lebih teratur dan jelas. Terdapat banyak model pengembangan perangkat lunak, dan cara untuk menentukan model mana yang sesuai untuk digunakan adalah dengan memahami domain permasalahan, *tools* yang akan digunakan, dan *deliverables* yang diperlukan. Penelitian ini menggunakan model pengembangan *waterfall* karena kebutuhan dari sistem yang akan dibangun memiliki kemungkinan kecil untuk berubah.

Pengembangan perangkat lunak model *waterfall* digunakan jika semua kebutuhan sudah diketahui dan dipahami sebelum pengembangan dimulai, dan memiliki kemungkinan yang kecil untuk berubah selama proses pengembangan perangkat lunak. Model ini merupakan contoh dari proses sekuensial yang berdasarkan perencanaan, sehingga pada prinsipnya, tim pengembang perangkat lunak sudah memiliki perencanaan dan jadwal untuk setiap aktivitas sebelum mulai mulai mengerjakan proyek (Sommerville, 2011).

Gambar 2.3 menggambarkan tahapan-tahapan dalam model *waterfall*, penjelasannya adalah sebagai berikut (Sommerville, 2011):

1. *Requirement Analysis and Definition*

   Tahapan ini terdiri dari proses analisis kebutuhan dan pendefinisian kebutuhan. Analisis kebutuhan dapat dilakukan dengan melakukan wawancara dengan klien, observasi, ataupun dengan analisis pada permasalahan yang akan diselesaikan. Analisis kebutuhan menghasilkan kebutuhan yang dapat didefinisikan dan dispesifikasikan, selain itu aktor yang berperan dalam sistem juga dapat teridentifikasi.



Gambar 2.2 Model *Waterfall*

Sumber: (Sommerville, 2011)

2. *System and Software Design*

Tahapan ini menggunakan hasil *requirement definition* untuk membuat representasi dari sistem yang akan dibangun. Perancangan yang dilakukan adalah perancangan arsitektur, perancangan data, perancangan komponen, dan perancangan antarmuka.

a. Perancangan arsitektur merepresentasikan struktur komponen yang membangun sistem (Pressman & Maxim, 2013). Perancangan arsitektur juga menggambarkan pertukaran pesan yang terjadi antar komponen.

b. Perancangan data merepresentasikan domain informasi yang dihasilkan dari analisis kebutuhan ke dalam struktur data (Pressman & Maxim, 2013). Perancanat data dilakukan dengan membuat *Conceptual Data Model* dan *Physical Data Model*. Pada level aplikasi, hasil perancangan data ditranslasikan ke dalam bentuk *database*.

c. Perancangan komponen merupakan representasi detail dari setiap komponen yang membangun sistem (Pressman & Maxim, 2013).

d. Perancangan antarmuka merepresentasikan bagaimana antarmuka sistem dengan pengguna. Perancangan ini dapat dilakukan dengan membuat *low-fidelity design*.

3. *Implementation and Unit Testing*

Implementasi merupakan tahapan dimana hasil perancangan direalisasikan ke dalam bentuk kode program sehingga dihasilkan sistem yang dapat digunakan. Pengujian unit juga dilakukan pada tahapan ini. Pengujian unit dilakukan untuk menguji unit terkecil dalam sistem (Pressman & Maxim, 2013), dalam pendekatam berorientasi objek unit terkecil adalah klas.

4. *Integration and System Testing*

Pengujian integrasi merupakan pengujian yang dilakukan pada *high level design*, yaitu hubungan antar unit-unit dalam sistem. Pengujian integrasi dilakukan dengan menggunakan teknik *white-box testing.* Pengujian sistem dapat dilakukan dengan dua teknik, yaitu *white-box testing* dan *black-box testing.*

a. *White-box Testing*

White-box testing merupakan pengujian yang memperharikan alur logika sistem. Kasus uji pada w*hite-box testing* dapat memastikan semua *independent path* dalam modul diuji paling tidak sekali, menguji semua alur pada kondisi benar dan salah, mengeksekusi semua perulangan pada batas-batas perulangan, dan menguji struktur data internal unutuk memastikan validitasnya (Pressman & Maxim, 2013).

b. *Black-box Testing*

Black-box testing disebut juga pengujian fungsional arau pengujian behavioral (Pressman & Maxim, 2013). Pengujian ini dilakukan tanpa memperhatikan logika internal sistem. *Black-box testing* dilakukan pada

fase akhir dalam pengembangan perangkat lunak, yatu ketika sistem sudah dapat digunakan.

5. Pemeliharaan

   Pemeliharaan meliputi perbaikan eror yang sebelumnya tidak diketahui, memperbarui implementasi dari unit sistem dan meningkatkan layanan sistem jika ditemukan kebutuhan baru.

2.3.2 Pendekatan Berorientasi Objek

Pengembangan perangkat lunak memiliki beberapa pendekatan, salah satunya adalah pendekatan berorientasi objek (*Object-Oriented*). Pendekatan berorientasi objek dapat diimplementasikan pada setiap tahap dalam model pengembangan perangkat lunak yang digunakan, dalam penelitian ini yang digunakan adalah model *waterfall*. Pada tahap analisis terdapat *Object-Oriented Analysis* (OOA), kemudian pada tahap perancangan dilakukan dengan *Object-Oriented Design* (OOD), dalam pembuatan kode program dilakukan dengan *Object-Oriented Programming* (OOP), dan terakhir, pengujian dapat dilakukan dengan *Object-Oriented Testing* (OOT).

Peranan utama dalam pendekatan berorientasi objek dipegang oleh objek. Objek sendiri dapat didefinisikan sebagai entitas dalam sistem yang mewakili suatu entitas di dunia nyata, misalnya adalah objek kucing, mobil, atau bunga. Objek-objek kemudian dapat diabstraksi menjadi klas-klas yang menyimpan data serta *behavior* dari objek-objek tersebut. Abstraksi ini merupakan salah satu konsep dasar dalam pemrograman berorientasi objek yang dapat membantu dalam meminimalisir kompleksitas perangkat lunak (Capretz, 2003).

Tahap pertama dalam pengembangan perangkat lunak dengan menggunakan pendekatan berorientasi objek adalah analisis berorientasi objek (OOA). OOA dilakukan untuk menganalisa kebutuhan, klas, dan juga objek yang diperlukan perangkat lunak berdasarkan domain permasalahannya. Untuk mempermudah pemahaman, kebutuhan sistem dapat dimodelkan dengan *use case diagram*.

Setelah kebutuhan perangkat lunak didapatkan, tahap selanjutnya adalah perancangan berorientasi objek (OOD). Perancangan dapat dilakukan pada level arsitektur, komponen, data, dan antarmuka. Pada tahap ini juga digunakan diagram-diagram untuk memodelkan perancangan perangkat lunak. Diagram yang dapat digunakan dalam perancangan di antaranya adalah *class diagram*, *sequence diagram*, *database schema*.

Rancangan perangkat lunak yang sudah dibuat kemudian diimplementasikan dengan pemrograman berorientasi objek (OOP). Konsep dasar dalam OOP adalah *abstraction* (abtraksi), *inheritance* (pewarisan), *encapsulation* (enkapsulasi), dan *polymorphism* (polimorfisme). Abstraksi merupakan cara pandang terhadap suatu entitas (objek) yang hanya memperhatikan hal-hal penting dan mengabaikan detail-detail yang tidak relevan dengan permasalahan, bentuk realisasi dari abstraksi merupakan klas. *Inheritance* atau pewarisan merupakan salah satu kunci pembeda sistem berorientasi objek dengan sistem konvensional. Jika suatu kelas

menjadi subklas dari suatu klas lain yang disebut superklas, maka subklas tersebut mewarisi semua atribut dan operasi yang berasosiasi dengan superklas. Hal ini berarti, semua struktur data dan algoritme yang mulanya dirancang dan diimplementasikan untuk superklas, dapat secara langsung tersedia untuk subklas tanpa perlu pekerjaan tambahan (Pressman & Maxim, 2013). Konsep pewarisan dalam OOP sangat meningkatkan tingkat penggunaan ulang komponen dari perangkat lunak (*reusability*). Untuk melengkapi konsep pewarisan, terdapat enkapsulasi, yaitu penyembunyian informasi detail dari suatu klas. Dua hal dasar dalam enkapsulasi adalah penyembunyian informasi dan antarmuka untuk mengakses data. Dengan menggunakan enkapsulasi, suatu klas dapat memiliki atribut atau *method* yang bersifat *private* atau publik. *Private* berarti atribut atau *method* hanya dapat digunakan di dalam klas, sedangkan publik berarti dapat digunakan oleh klas lain. Polimorfisme merupakan konsep dimana terdapat beberapa klas dengan *signature method* yang sama, namun implementasinya berbeda berdasarkan setiap klas (Andre, 2014). OOP dapat dilakukan dengan menggunakan bahasa pemrograman berorientasi objek, misalnya adalah Java, SmallTalk, C++, ObjectiveC.

Hasil implementasi kode program kemudian diuji dengan pegujian berorientasi objek (OOT). Pengujian dapat menggunakan teknik pengujian *black-box* atau *white-box*. OOT, yang disebut unit merupakan klas atau objek, sehingga setiap klas yang ada pada klas turunan pun harus diperiksa. Sedangkan pengujian validasi berfokus pada aksi pengguna dan hasil yang diterima pengguna.

2.3.3 Pemodelan Berorientasi Objek

Standar *de facto* untuk pemodelan berorientasi objek adalah UML (*Unified Modeling Language*) (Sommerville, 2011). UML pertama kali muncul pada tahun 1990 dan revisi mayornya selesai pada tahun 2004 dan menghasilkan UML 2 yang kini digunakan secara luas (Sommerville, 2011). Pemodelan ini dilakukan untuk memberikan ilustrasi dalam bentuk diagram untuk mempermudah pemahaman pengguna, pemangku kepentingan, maupun tim pengembang perangkat lunak.

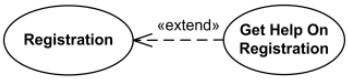
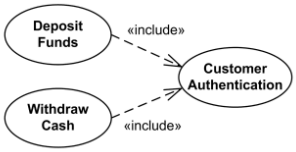
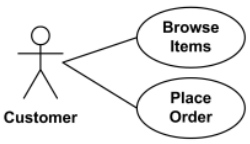
Sommerville (2011) mengutip survey pada tahun 2007 yang dilakukan oleh Erickson dan Siau yang menunjukkan bahwa terdapat lima diagram UML yang dapat merepresentasikan esensi sistem, yaitu; (1) *Activity diagram*, menggambarkan aktifitas-aktifitas yang terlibat dalam proses atau pemrosesan data, (2) *Use case diagram*, menggambarkan interaksi antara sistem dan lingkungannya, (3) *Sequence diagram,* menampilkan interaksi antara aktor dengan sistem dan juga antara komponen sistem, (4) *Class diagram*, menggambarkan klas-klas dari objek yang terdapat pada sistem dan bagaimana hubungan di antaranya, dan (5) *State diagram*, menampilkan bagaimana sistem bereksi terhadap *event* internal atau eksternal.

Penelitian ini menggunakan *use case diagram* untuk memodelkan kebutuhan sistem serta *class diagram* dan *sequence diagram* untuk perancangan arsitektur. Penjelasan mengenai ketiga diagram yang digunakan adalah sebagai berikut:

1. *Use case diagram*

Use case diagram (diagram *use case*) digunakan secara luas untuk memodelkan kebutuhan sistem pada tahapan analisis. Diagram ini menggambarkan interaksi antara sistem dan lingkungannya dari sudut pandang aktor. Pada Tabel 2.3 berikut merupakan simbol dan notasi yang terdapat dalam diagram *use case.*

Tabel 2.3 Simbol dan notasi pada diagram *use case*

Nama simbol/notasi	Gambar	Deskripsi
Aktor		Aktor merepresentasikan pengguna atau subsistem lain yang berinteraksi dengan sistem.
Use case		*Use case* merepresentasikan aksi yang dapat dilakukan aktor. *Use case* digambarkan dengan bentuk *eclipse* dengan nama di dalamnya.
Extend		*Extend* merupakan hubungan langsung yang digunakan apabila ada suatu *use case* yang berhubungan dengan use case lain yang bersifat opsional. *Use case* yang di-*extend* dapat berdiri sendiri secara independen.
Include		*Include* merepresentasikan hubungan langsung antara dua *use case* yang hubungan ini bersifat '*required*' atau diperlukan, bukan bersifat opsional.
Asosiasi		Asosiasi merepresentasikan komunikasi antara aktor dan *use case*.

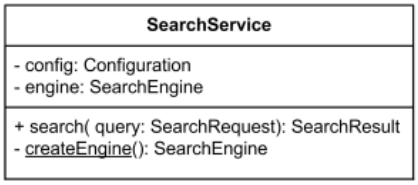
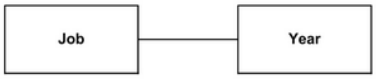
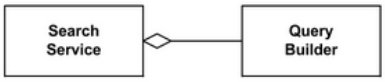
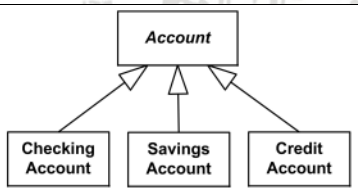
Sumber: (Fowler, 2003)

2. *Class diagram*

Class diagram (diagram klas) merupakan diagram UML yang menggambarkan klas-klas dari objek yang terdapat pada sistem dan bagaimana hubungan di antaranya. Diagram klas menampilkan struktur dari klas-klas beserta atribut,

operasi (*method*), dan juga hubungannya dengan klas lain. Pada Tabel 2.4 berikut ini merupakan simbol dan notasi dasar pada diagram klas.

Tabel 2.4 SImbol dan notasi pada diagram klas

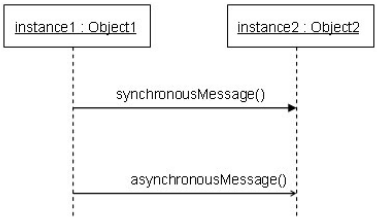
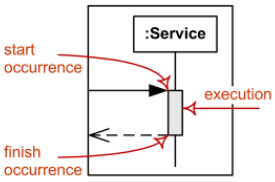
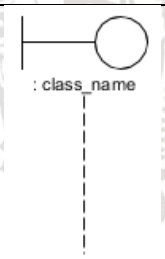
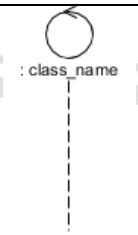
Nama simbol/notasi	Gambar	Deskripsi
Klas		Notasi klas dapat dibagi menjadi kompartemen pertama, yang paling atas merupakan nama klas, kemudian kompartemen tengah merupakan atribut-atribut, dan kompartemen terakhir adalah operasi atau *method* pada klas tersebut.
Asosiasi		Asosiasi merupakan hubungan antaran klas yang digunakan untuk mengilustrasikan instan dari klas tersebut bisa saling berhubungan.
Agregasi		Agregasi dinotasikan dengan asosiasi binary dengan *dimond* kosong pada akhir garis asosiasi agregat.
Generalisasi		Generalisasi dinotasikan dengan garis yang memiliki kepala panah segitiga mengarah kepada klas yang bersifat general.

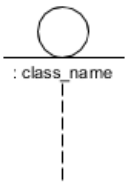
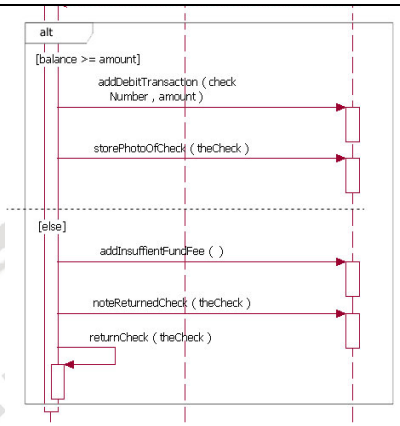
Sumber: (Fowler, 2003)

3. *Sequence diagram*

Sequence diagram berfungsi untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antara objek dalam sistem yang digambarkan berdasarkan waktu. Beberapa simbol dan notasi pada diagram ini dapat dilihat pada Tabel 2.5.

Tabel 2.5 Simbol dan notasi pada diagram *sequence*

Nama simbol/notasi	Gambar	Deskripsi
Lifeline		*Lifeline* dinotasikan dengan garis putus-putus yang ditarik dari bawah objek hingga waktu objek tersebut tidak lagi hidup atau diperlukan.
Messages		*Messages* merupakan bentuk interaksi antar objek yang menggambarkan pesan antara objek tersbeut. Pesan ini dapat menggambarkan pemicu aktifnya suatu operasi atau *trigger* aktivitas pada suatu objek.
Activation Boxes		*Activation boxes* direpresentasikan dengan persegi panjang vertikal yang mendefinisikan waktu mulai hingga selesai eksekusi suatu operasi atau aksi dari suatu objek.
Boundary		*Boundary* merupakan representasi objek antarmuka pada sistem.
Controller		*Controller* merupakan representasi objek pusat logika pada sistem.

Nama simbol/notasi	Gambar	Deskripsi
Entity		*Entity* merupakan representasi entitas pada basis data sistem.
Alternative		*Alternative* digunakan ketika ada dua atau lebih urutan pesan yag dapat terjadi dalam sistem. *Alternative* memungkinkan pemodelan logika kondisional "*if then else*". Notasi *alternative* yaitu dengan menggunakan *frame* dengan tulisan "*alt*".

Sumber: (Fowler, 2003)

2.4 Teknologi Pengembangan Perangkat Lunak

Pengembangan aplikasi penentuan prioritas kebutuhan fungsional berdasarkan kebutuhan non-fungsional dibangun pada *platform web* dengan menggunakan bahasa pemrograman PHP dengan *framework* CodeIgniter. Sedangkan untuk bagian *front-end* digunakan bahasa pemrograman HTML, javascript, dan CSS dengan *framework* MaterializeCSS.

2.4.1 CodeIgniter

CodeIgniter merupakan salah satu *framework* sumber terbuka untuk bahasa pemrograman PHP. CodeIgniter menggunakan arsitektur MVC (*Model View Controller*) yang dikembangkan oleh Taylor Otwell dengan tujuan untuk mengurangi biaya awal pengembangan dan juga meningkatkan kualitas kode program dengan mendefinisikan praktik desain standar industri (Nilanchala, 2017). Arsitektur MVC pada CodeIgniter sesuai untuk pengembangan *web* berskala besar dengan memisahkan masukkan, proses, dan keluaran dari aplikasi ke dalam tiga objek yaitu *model*, *view*, dan *controller*. Pemisahan ini membuat setiap objek dapat melaksanakan dan menyelesaikan tugasnya masing-masing dan tidak saling memperngaruhi apabila ada salah satu objek yang berubah (Zhang & Liu, 2017). *Framework* CodeIgniter digunakan agar kode program memiliki struktur yang lebih rapi dan memudahkan pengembangan lanjut atau pemeliharaan sistem.

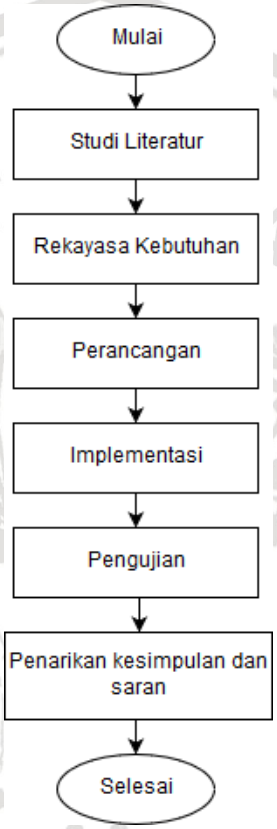
2.4.2 MaterializeCSS

MaterializeCSS merupakan *front-end framework* yang menggunakan konsep Material Design dari Google. MaterializeCSS dikembangkan untuk mendukung pegembangan web yang responsif dengan desain yang modern. *Framework* ini dibuat dengan menggunakan HTML, CSS, dan JavaScript.

MaterializeCSS, dapat digunakan dengan dua cara, yaitu instalasi lokal dan CDN *based version*. Instalasi lokal bisa dilakukan dengan mengunduh file materialize.min.css dan materialize.min.js, kemudian menempatkannya di direktori website dan menyertakannya ke dalam kode HTML. CDN *based version* yaitu menyertakan langsung file materialize.min.css dan materialize.min.js pada kode HTML langsung dari *Content Delivery Network* (CDN). MaterializeCSS memudahkan pengembang *web* jika ingin membangun situs web parallax dan juga pengembangan situs web dengan menggunakan Material Design dari Google untuk desain yang modern dan responsif.

BAB 3 METODOLOGI PENELITIAN

Bab ini menjelaskan langkah-langkah yang dilakukan dalam penelitian. Penelitian yang dilakukan penulis merupakan penelitian implementatif, yaitu pengembangan (*development*). Langkah-langkah yang dilakukan dalam penelitian ini adalah studi literatur, kemudian pengembangan aplikasi dengan menggunakan SDLC model *waterfall* yang terdiri dari analisis kebutuhan, perancangan, implementasi, dan pengujian, serta terakhir dilakukan pengambilan kesimpulan dari hasil penelitian. Diagram alir dari metodologi penelitian ini dapat dilihat pada gambar 3.1 di bawah ini.



Gambar 3.1 Diagram alir metodologi penelitian

3.1 Studi Literatur

Studi literatur dilakukan untuk mendapatkan pengetahuan lebih dalam mengenai teori-teori yang diperlukan dalam pengembangan aplikasi penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional. Studi literatur ini dilakukan dengan membaca hasil penelitian sebelumnya yang berkaitan dengan penelitian yang dilakukan serta mengenai teknologi yang diperlukan dalam pengembangan aplikasi dalam penelitian ini. Sumber literatur berupa jurnal, prosiding konferensi, buku, dan dokumentasi. Pokok bahasan literatur meliputi:

1. Kajian pustaka mengenai penelitian terkait

2. Kebutuhan perangkat lunak
    a. Proses rekayasa kebutuhan
    b. Metode penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional
3. Pengembangan perangkat lunak
    a. Model pengembangan perangkat lunak
    b. Pendekatan berorientasi objek
    c. Pemodelan berorientasi objek
4. Teknologi yang digunakan dalam pengembangan perangkat lunak
    a. CodeIgniter
    b. MaterializeCSS

3.2 Rekayasa Kebutuhan

Rekayasa kebutuhan terdiri elisitasi kebutuhan, deskripsi umum sistem, identifikasi aktor, dan pemodelan kebutuhan. Elisitasi kebutuhan dilakukan untuk mendapatkan kebutuhan-kebutuhan yang harus disediakan sistem dan dijelaskan mengenai gambaran umum sistem. Elisitasi kebutuhan dilakukan dengan menganalisis penelitian penentuan prioritas kebutuhan yang dilakukan oleh Umang Garg dan Abhishek Singhal (Garg & Singhal, 2017) yang berjudul "*Software Requirement Prioritization Based on Non-Functional Requirement*". Hasil dari analisis kebutuhan adalah identifikasi aktor sistem, definisi dan spesifikasi kebutuhan, dan pemodelan kebutuhan berupa *use case diagram* dan *use case scenario*. Deskripsi umum sistem menjelaskan mengenai sistem yang dikembangkan berdasarkan hasil elisitasi kebutuhan.

3.3 Perancangan

Perancangan dilakukan berdasarkan hasil analisis kebutuhan dan digunakan sebagai acuan untuk implementasi sistem. Perancangan dilakukan dengan pendekatan berorientasi objek, sehingga pemodelan dilakukan dengan UML (*Unified Modelling Language*). Perancangan yang dibuat adalah sebagai berikut:

a. Perancangan Arsitektur

Perancangan arsitektur berkaitan dengan pemahaman mengenai bagaimana organisasi dan keseluruhan struktur pada sistem (Sommerville, 2011). Dalam pendekatan berorientasi objek, perancangan arsitektur bisa dilakukan dengan menggunakan *sequence diagram* dan *class diagram*. *Sequence diagram* menggambarkan interaksi antara aktor dengan sistem dan juga antara komponen-komponen dalam sistem (Sommerville, 2011). Pesan-pesan yang dikirim atau diterima setiap objek dapat digambarkan melalui diagram ini. *Class diagram* merupakan diagram yang menunjukan objek klas-klas yang ada dalam sistem dam hubungan di antaranya (Sommerville, 2011).

b. Perancangan Data

Perancangan data mentransformasikan data-data yang didapat pada rekayasa kebutuhan sehingga didapatkan struktur daya. Perancangan data dilakukan dengan membuat *Conceptual Data Model* (CDM) dan *Physical Data Model* (PDM). CDM menghasilkan entitas, atribut, dan relasi di antara entitas yang ada. CDM direpresentasikan dalam bentuk *Entity Realtionship Diagram* (ERD). PDM merupakan perancangan yang lebih mendetail dan mendekati struktur *database*.

c. Perancangan Komponen

Perancangan komponen merupakan representasi detail dari setiap komponen sistem dalam bentuk yang mudah dipahami manusia. Komponen pembangun sistem dalam pendekatan berorientasi objek merupakan klas, sehingga dalam perancangan komponen ini diambil tiga sampel *method* yang berasal dari klas-klas yang ada. Perancangan komponen dilakukan dengan membuat algoritme dalam bentuk *pseudocode*.

d. Perancangan Antarmuka

Perancangan antarmuka merupakan rancangan dari tata letak komponen-komponen dalam sistem yang dibangun. Perancangan antarmuka dibuat dengan membuat rancangan *low-fidelity*, yaitu dengan membuat *wireframe* yang menggambarkan tata letak komponen-komponen pada antarmuka sistem.

3.4 Implementasi

Implementasi merupakan tahap untuk merealisasikan rancangan yang sudah dibuat ke dalam bentuk kode program sehingga menjadi sistem yang dapat berfungsi. Implementasi sistem dilakukan pada *platform web*. Spesifikasi sistem yang digunakan dalam proses implementasi merupakan salah satu hal yang dijelaskan dalam tahap ini. Implementasi yang dilakukan adalah implementasi kode program, implementasi data, dan implementasi antarmuka. Implementasi kode program dilakukan dengan menggunakan *framework* CodeIgniter, untuk implementasi antarmuka digunakan *framework* MaterializeCSS, sedangkan implementasi data menggunakan MySQL.

3.5 Pengujian

Pengujian dilakukan untuk memastikan bahwa sistem yang dibangun sudah sesuai dengan kebutuhan yang telah dianalisis dan bekerja dengan benar dan menghasilkan keluaran yang benar. Pengujian sistem dilakukan dengan pengujian unit, pengujian integrasi, dan pengujian validasi.

Pengujian unit dilakukan dengan teknik pengujian *white box* yang dilakukan pada unit terkecil dalam sistem, yaitu klas. Pengujian unit dilakukan dengan mengambil tiga sampel *method* dari klas yang diuji dan dilakukan pengujian dengan metode *basis path testing*. Pengujian integrasi dilakukan pada *high level design*, yaitu hubungan antar unit-unit dalam sistem. Pengujian integrasi dilakukan dengan teknik pengujian *white box* dan metode *basis path testing*. Pengujian

validasi dilakukan dengan menggunakan teknik pengujian *black box*. Tujuan dari pengujian ini adalah memastikan seluruh kebutuhan sistem sudah terpenuhi dan sistem dapat memberikan hasil keluaran sesuai yang diharapkan.

3.6 Penarikan Kesimpulan

Penarikan kesimpulan dilakukan setelah seluruh tahap penelitian selesai dilaksanakan. Kesimpulan yang diambil merupakan hasil analisis dari keseluruhan kinerja sistem yang dikembangkan dan akan menghasilkan kesimpulan yang menjawab rumusan masalah yang telah dipaparkan di awal. Selain kesimpulan, terdapat juga saran sebagai catatan untuk pengembangan sistem yang mungkin dilakukan di masa yang akan datang.

BAB 4 REKAYASA KEBUTUHAN

Rekayasa kebutuhan merupakan tahapan untuk menggali kebutuhan sistem sehingga didapat kebutuhan yang harus disediakan sistem untuk mencapai tujuan pengguna. Selain menjelaskan kebutuhan sistem, analisis kebutuhan juga menjelaskan proses elisitasi yang dilakukan dan identifikasi aktor yang berperan dalam sistem.

4.1 Elisitasi Kebutuhan Sistem

Elisitasi kebutuhan sistem dilakukan dengan melakukan analisis terhadap metode pada *paper* "*Software Requirement Prioritization Based on Non-Functional Requirement*" yang ditulis oleh Umang Garg dan Abhishek Shinghal.

Kebutuhan perangkat lunak yang diprioritaskan dalam *paper* Garg dan Shinghal adalah kebutuhan fungsional dengan berdasarkan pada kebutuhan non-fungsional. Proses penentuan prioritas kebutuhan perangkat lunak dilakukan oleh analis. Langkah pertama yang dilakukan adalah melakukan perbandingan secara berpasangan (*pair-wise comparison*) antara kebutuhan non-fungsional serta menentukan kebutuhan yang bersifat lebih dominan dari kedua kebutuhan yang dibandingkan beserta nilai yang berdasarkan skala Saaty pada Tabel 2.1. Nilai *pair-wise comparison* bersifat subjektif berdasarkan pendapat dan pengalaman analis.

Misal terdapat 3 buah kebutuhan non-fungsional, yaitu *security, data integrity,* dan *usability*. Kebutuhan yang sama akan memiliki nilai perbandingan 1, karena nilai kepentinggannya tentu sama. Kemudian, misal antara *security* dan *data integrity*, dianggap *security* lebih penting dengan nilai skala perbandingan yang sesuai adalah 3 (*security* dianggap sedikit lebih penting dari *data integrity*), maka didapat juga untuk perbandingan antara *data integrity* terhadap *security* nilainya adalah 1/3. Perbandingan yang dilakukan adalah sebanyak jumlah kebutuhan yang ada. Dan dari perbandingan yang telah dilakukan akan terbentuk matriks perbandingan seperti pada Tabel 4.1.

Tabel 4.1 Studi Kasus *Pair-wise Comparison* Kebutuhan Non-Fungsional

	Security	*Data Integrity*	*Usability*
Security	1	3	1/5
Data Integrity	1/3	1	¼
Usability	5	4	1
Total	19/3	8	29/20

Setelah didapatkan matriks perbandingan seperti pada Tabel 4.1, langkah yang dilakukan selanjutnya adalah menghitung nilai prioritas dari masing-masing kebutuhan dengan cara menghitung jumlah dari pembagian nilai perbandingan

pada setiap kolom dengan jumlah kolom dengan menggunakan rumus pada Persamaan 2.1. Hasilnya dalam bentuk tabel adalah seperti berikut ini:

Tabel 4.2 Studi Kasus Penentuan Prioritas Kebutuhan Non-Fungsional

	Security	*Data Integrity*	*Usability*	Prioritas
Security	3/19	3/8	4/29	0.671
Data Integrity	1/19	1/8	5/29	0.35
Usability	15/19	½	20/29	1.978

Setelah mendapatkan nilai prioritas dari setiap kebutuhan non-fungsional, maka selanjutnya dilakukan perbandingan antara kebutuhan fungsional dengan non-fungsional menggunakan skala pada Tabel 2.2.

Misal terdapat 2 kebutuhan fungsional, yaitu:

1. Sistem harus menyediakan fungsi *login* dengan estimasi waktu untuk mengimplementasikan kebutuhan adalah 1 hari dan estimasi biayanya 100.000.
2. Sistem harus menyediakan fungsi *logout* dengan estimasi waktu untuk mengimplementasikan kebutuhan adalah 1 hari dan estimasi biayanya 50.000.

Setiap kebutuhan fungsional di atas kemudian dibandingkan dengan kebutuhan non-fungsional dalam bentuk matriks seperti pada Tabel 4.3.

Tabel 4.3 Studi Kasus Penentuan Prioritas Kebutuhan Non-Fungsional

	Security	*Data Integrity*	*Usability*
Sistem harus menyediakan fungsi *login*	3	1	2
Sistem harus menyediakan fungsi *logout*	5	3	1

Nilai perbandingan yang sudah didapatkan dalam Tabel 4.3 kemudian dikalikan dengan nilai prioritas dari setiap kebutuhan non-fungsional yang bersangkutan dengan menggunakan rumus seperti pada Persamaan 2.2. Hasil perkalian tersebut kemudian dijumlahkan agar didapat nilai prioritas dari setiap kebutuhan fungsional.

Tabel 4.4 Studi Kasus Penentuan Prioritas Kebutuhan Non-Fungsional

	Security	*Data Integrity*	*Usability*	Prioritas	Waktu	Waktu Kumulatif	Biaya	Biaya Kumulatif

Sistem harus menyediakan fungsi *login*	(3 * 0.671) = 2.013	(1 * 0.35) = 0.35	(2 * 1.978) = 3.956	6.319	1	1	100.000	100.000
Sistem harus menyediakan fungsi *logout*	(5 * 0.671) = 3.355	(3 * 0.35) = 1.05	(1 * 1.978) = 1.978	6.383	1	2	50.000	150.000

Estimasi waktu dan estimasi biaya implementasi diperlukan untuk mengetahui berapa lama waktu dan biaya total yang diperlukan untuk mengimplementasikan kebutuhan-kebutuhan yang ada, dan juga untuk menyesuaikan antara waktu dan biaya yang dimiliki dengan waktu dan biaya yang diperlukan setiap kebutuhan. Namun, estimasi waktu dan biaya ini tidak mempengaruhi prioritas dari kebutuhan.

Penentuan prioritas kebutuhan hanya bisa dilakukan apabila daftar kebutuhan sudah dianggap lengkap, karena prosesnya memerlukan semua kebutuhan untuk dibandingkan, sehingga apabila ada kebutuhan yang baru, perhitungan prioritas kebutuhan harus dilakukan dari awal. Prioritas yang dihasilkan disini adalah rekomendasi urutan dalam mengimplementasikan kebutuhan yang ada berdasarkan hasil perhitungan yang dilakukan.

4.2 Gambaran Umum Sistem

Aplikasi penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional merupakan sebuah aplikasi berbasis *web* yang diperuntukkan bagi analis untuk membantu proses penentuan prioritas kebutuhan fungsional. Aplikasi ini memiliki fungsi utama untuk menentukan prioritas kebutuhan fungsional perangkat lunak. Fungsi lain yang disediakan adalah menambah kebutuhan fungsional dan menambah kebtuhan non-fungsional yang berfungsi untuk menyimpan kebutuhan yang diperlukan ke dalam *database.* Terdapat juga fungsi untuk menampilkan daftar kebutuhan fungsional, menampilkan daftar kebutuhan non-fungsional yang menyediakan informasi dari data kebutuhan yang sudah ada, mengubah kebutuhan fungsional, mengubah kebutuhan non-fungsional, menghapus kebutuhan fungsional, dan menghapus kebutuhan non-fungsional.

Penentuan prioritas kebutuhan fungsional dilakukan berdasarkan kebutuhan non-fungsional, sehingga sebelum menentukan prioritas kebutuhan fungsional harus dilakukan penentuan prioritas kebutuhan non-fungsional terlebih dahulu. Aplikasi memberikan rekomendasi urutan prioritas kebutuhan dalam bentuk

daftar kebutuhan yang diurutkan dari kebutuhan dengan nilai prioritas tertinggi hingga terendah berdasarkan proses penentuan prioritas yang sudah dilakukan.

4.3 Identifikasi Aktor

Aktor yang berperan dalam penentuan prioritas kebutuhan berdasarkan metode yang dijelaskan pada *paper* Garg dan Shinghal adalah sistem analis. Sistem analis berperan dalam memberi nilai kepentingan pada proses perbandingan berpasangan baik dalam perhitungan prioritas kebutuhan non-fungsional maupun pada proses perhitungan fungsional. Sistem analis kemudian disebut sebagai analis dalam penulisan penelitian ini. Tabel 4.1 berikut merupakan daftar identifikasi aktor beserta deskripsinya.

Tabel 4.5 Identifikasi Aktor

No	Aktor	Deskripsi
1	Analis	Analis merupakan seorang individu atau kelompok yang merupakan bagian dari tim pengembangan perangkat lunak yang berperan dalam mengidentifikasi kebutuhan pengguna atau klien serta tujuan bisnis dari dikembangkannya suatu sistem. Analis memiliki peran untuk menyatakan secara spesifik kebutuhan-kebutuhan yang harus disediakan sistem untuk mencapai tujuan pengguna sistem.

4.4 Daftar Kebutuhan Fungsional

Kebutuhan fungsional adalah deskripsi layanan, fungsi, ataupun fitur yang disediakan sistem untuk penggunanya (Siahaan, 2012). Kebutuhan fungsional dalam penelitian ini akan diberi penomoran RPBN-F-XX-YY, dengan RPBN merupakan nama sistem yaitu *Requirement Prioritization Based on Non-Functional*, F menunjukkan bahwa kebutuhan merupakan kebutuhan fungsional, XX merupakan nomor kebutuhan, dan YY merupakan nomor spesifikasi kebutuhan.

Tabel 4.6 berikut ini merupakan pendefinisian kebutuhan fungsional sistem beserta spesifikasinya, aktor yang bersangkutan, dan pemetaan nama dalam *use case* yang didapatkan dari proses analisis yang dilakukan terhadap metode yang dijelaskan pada *paper* Garg dan Shinghal.

Tabel 4.6 Definisi dan Spesifikasi Kebutuhan Fungsional dan Pemetaan Nama *Use Case*

No	Kode Kebutuhan	Definisi Kebutuhan	Aktor	Nama *Use Case*
1	RPBN-F-01	Sistem harus menyediakan fungsi untuk menambah kebutuhan fungsional.	Analis	Menambah kebutuhan fungsional

1.1	RPBN-F-01-01	Sistem harus menyediakan tombol "Tambah kebutuhan Fungsional" sebagai sarana untuk menambah kebutuhan non-fungsional.		
1.2	RPBN-F-01-02	Sistem harus menyediakan *form* tambah kebutuhan yang terdiri dari kolom isian kode kebutuhan, deskripsi kebutuhan, estimasi waktu untuk implementasi kebutuhan, dan estimasi biaya untuk implementasi kebutuhah.		
1.3	RPBN-F-01-03	Kolom isian deskripsi kebutuhan hanya dapat diisi dengan karakter alfanumerik dan juga simbol.		
1.4	RPBN-F-01-04	Kolom isian estimasi waktu untuk implementasi kebutuhan dan estimasi biaya untuk implementasi kebutuhah hanya bisa diisi oleh karakter numerik.		
1.5	RPBN-F-01-05	Sistem harus menyediakan tombol "Tambah" untuk menyimpan kebutuhan fungsional.		
1.6	RPBN-F-01-06	Sistem harus menyediakan tombol "Batal" untuk membatalkan penambahan kebutuhan fungsional.		
1.7	RPBN-F-01-07	Sistem harus menampilkan pesan peringatan jika ada kolom isian yang belum diisi.		
2	RPBN-F-02	Sistem harus menyediakan fungsi untuk menambah kebutuhan non-fungsional.	Analis	Menambah kebutuhan non-fungsional

2.1	RPBN-F-02-01	Sistem harus menyediakan tombol "Tambah Kebutuhan Non-Fungsional" sebagai sarana untuk menambahkan kebutuhan non-fungsional.		
2.2	RPBN-F-02-02	Sistem harus menyediakan *form* tambah kebutuhan non-fungsional yang terdiri dari kolom isian kode kebutuhan dan deskripsi kebutuhan.		
2.3	RPBN-F-02-03	Kolom isian deskripsi kebutuhan hanya dapat diisi dengan karakter alfanumerik dan juga simbol.		
2.4	RPBN-F-02-04	Sistem harus menyediakan tombol "Tambah" untuk menyimpan kebutuhan non-fungsional.		
2.5	RPBN-F-02-05	Sistem harus menyediakan tombol "Batal" untuk membatalkan penambahan kebutuhan non-fungsional.		
2.6	RPBN-F-02-06	Sistem harus menampilkan pesan peringatan jika ada kolom isian yang belum diisi.		
3	RPBN-F-03	Sistem harus menyediakan fungsi untuk menampilkan daftar kebutuhan fungsional.	Analis	Melihat daftar kebutuhan fungsional
3.1	RPBN-F-03-01	Data yang ditampilkan adalah kode kebutuhan, deskripsi kebutuhan, estimasi waktu, waktu kumulatif, estimasi biaya, biaya kumulatif, dan prioritas kebutuhan.		
3.2	RPBN-F-03-02	Sistem menampilkan kebutuhan fungsional		

		secara berurutan dari kebutuhan dengan prioritas tertinggi hingga terendah.		
4	RPBN-F-04	Sistem harus menyediakan fungsi untuk menampilkan daftar kebutuhan non-fungsional.	Analis	Melihat daftar kebutuhan non-fungsional
4.1	RPBN-F-04-01	Data yang ditampilkan adalah kode kebutuhan, deskripsi kebutuhan, dan prioritas kebutuhan.		
4.2	RPBN-F-04-02	Sistem menampilkan kebutuhan fungsional secara berurutan dari kebutuhan dengan prioritas tertinggi hingga terendah.		
5	RPBN-F-05	Sistem harus menyediakan fungsi untuk mengubah data pada kebutuhan fungsional yang ada pada daftar kebutuhan.	Analis	Mengubah kebutuhan fungsional
5.1	RPBN-F-05-01	Sistem harus menyediakan *form* perubahan data kebutuhan fungsional.		
5.2	RPBN-F-05-02	Kolom isian deskripsi kebutuhan pada *form* perubahan data kebutuhan fungsional hanya dapat diisi dengan karakter alfanumerik dan simbol.		
5.3	RPBN-F-05-03	Kolom isian estimasi waktu implementasi kebutuhan pada *form* perubahan data kebutuhan fungsional hanya dapat diisi dengan karakter numerik.		
5.4	RPBN-F-05-04	Kolom isian estimasi biaya implementasi kebutuhan pada *form* perubahan data kebutuhan fungsional		

		hanya dapat diisi dengan karakter numerik.		
5.5	RPBN-F-05-05	Sistem harus menyediakan tombol "Simpan" untuk menyimpan perubahan data kebutuhan fungsional.		
5.6	RPBN-F-05-06	Sistem harus menyediakan tombol "Batal" untuk membatalkan perubahan data kebutuhan fungsional.		
5.7	RPBN-F-05-07	Sistem harus menampilkan pesan peringatan jika ada kolom isian yang kosong.		
6	RPBN-F-06	Sistem harus menyediakan fungsi untuk mengubah data pada kebutuhan non-fungsional yang ada pada daftar kebutuhan.	Analis	Mengubah kebutuhan non-fungsional
6.1	RPBN-F-06-01	Sistem harus menyediakan *form* perubahan data kebutuhan non-fungsional.		
6.2	RPBN-F-06-02	Kolom deskripsi kebutuhan pada *form* perubahan data kebutuhan non-fungsional hanya dapat diisi dengan karakter alfanumerik dan simbol.		
6.3	RPBN-F-06-03	Sistem harus menyediakan tombol "Simpan" untuk menyimpan perubahan data kebutuhan non-fungsional.		
6.4	RPBN-F-06-04	Sistem harus menyediakan tombol "Batal" untuk membatalkan perubahan data kebutuhan non-fungsional.		
6.5	RPBN-F-06-05	Sistem harus menampilkan pesan peringatan jika ada kolom isian yang kosong.		

7	RPBN-F-07	Sistem harus menyediakan fungsi untuk menghapus kebutuhan yang sudah ada dalam daftar kebutuhan.	Analis	Menghapus kebutuhan fungsional
7.1	RPBN-F-07-01	Sistem harus menyediakan tombol hapus sebagai sarana untuk menghapus kebutuhan.		
7.2	RPBN-F-07-02	Sistem harus menampilkan dialog konfirmasi penghapusan jika tombol hapus diklik.		
7.3	RPBN-F-07-03	Sistem harus menyediakan tombol “Ya” pada dialog konfirmasi untuk konfirmasi penghapusan kebutuhan.		
7.4	RPBN-F-07-04	Sistem harus menyediakan tombol “Tidak” pada dialog konfirmasi untuk membatalkan penghapusan kebutuhan.		
8	RPBN-F-08	Sistem harus menyediakan fungsi untuk menghapus kebutuhan yang sudah ada dalam daftar kebutuhan.	Analis	Menghapus kebutuhan non-fungsional.
8.1	RPBN-F-08-01	Sistem harus menyediakan tombol hapus sebagai sarana untuk menghapus kebutuhan.		
8.2	RPBN-F-08-02	Sistem harus menampilkan dialog konfirmasi penghapusan jika tombol hapus diklik.		
8.3	RPBN-F-08-03	Sistem harus menyediakan tombol “Ya” pada dialog konfirmasi untuk konfirmasi penghapusan kebutuhan.		
8.4	RPBN-F-08-04	Sistem harus menyediakan tombol “Tidak” pada dialog		

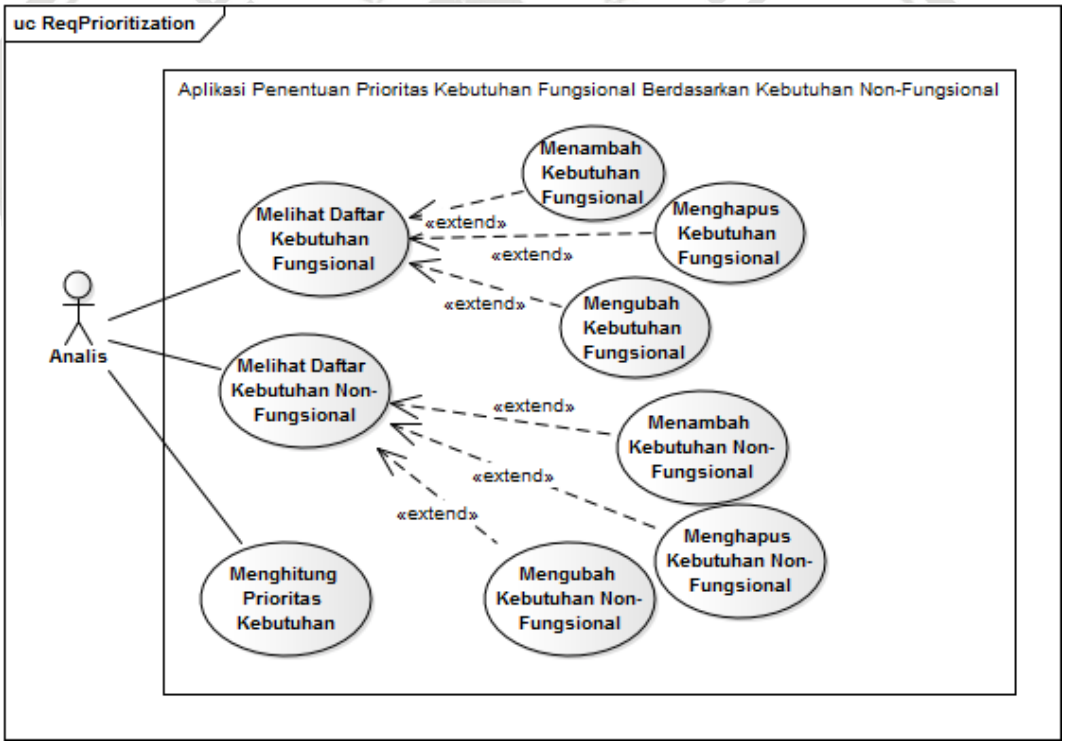
		konfirmasi untuk membatalkan penghapusan kebutuhan.		
9	RPBN-F-09	Sistem harus menyediakan fungsi untuk perhitungan prioritas kebutuhan.	Analis	Menghitung prioritas kebutuhan
9.1	RPBN-F-09-01	Sistem harus menyediakan informasi skala yang digunakan dalam perhitungan prioritas kebutuhan non-fungsional dan perhitungan prioritas fungsional.		
9.2	RPBN-F-09-02	Sistem harus menyediakan daftar kebutuhan non-fungsional beserta kebutuhan yang akan dibandingkan.		
9.3	RPBN-F-09-03	Sistem harus menyediakan kolom isian nilai kepentingan kebutuhan non-fungsional hanya bisa diisi karakter numerik 1-9.		
9.4	RPBN-F-09-04	Sistem harus menyediakan sarana untuk memilih kebutuhan non-fungsional yang dianggap lebih dominan.		
9.5	RPBN-F-09-05	Sistem harus menyediakan daftar kebutuhan fungsional beserta kebutuhan non-fungsional yang akan dibandingkan.		
9.6	RPBN-F-09-06	Sistem harus menyediakan kolom isian nilai kepentingan kebutuhan fungsional hanya bisa diisi karakter numerik 1-5.		
9.7	RPBN-F-09-07	Sistem harus menampilkan pesan peringatan jika ada kolom isian nilai		

		kepentingan yang belum diisi.		
9.8	RPBN-F-09-08	Sistem harus menampilkan hasil perhitungan prioritas kebutuhan secara berurutan dari prioritas tertinggi ke prioiritas terendah.		

4.5 Pemodelan Kebutuhan

4.5.1 *Use Case Diagram*

Use case diagram merupakan ilustrasi visual dari interaksi aktor dengan sistem dari sudut pandang aktor. *Use case diagram* dibuat berdasarkan kebutuhan fungsional yang sudah dianalisis untuk mempermudah pemahaman fungsionalitas sistem. *Use case diagram* dari sistem yang akan dibuat dalam penelitian ini bisa dilihat pada Gambar 4.1.



Gambar 4.1 *Use Case Diagram* Sistem

4.5.2 *Use Case Scenario*

Use case scenario merupakan bentuk tekstual yang menjelaskan lebih detail interaksi antara sistem dan aktor. Setiap skenario menjelaskan secara lengkap tujuan dari *use case*, aktor yang berperan dalam *use case* tersebut, kondisi yang harus dipenuhi untuk menjalankan *use case* (*pre-condition*), alur utama *use case*

(main flow), alur alternatif (*alternative flows*), dan juga kondisi akhir setelah *use case* deksekusi (*post-condition*).

Tabel 4.7 sampai dengan Tabel 4.16 berikut merupakan *use case scenario* pada sistem yang akan dikembangkan.

Tabel 4.7 *Use case scenario* untuk Menambah Kebutuhan Fungsional

Flow of Events untuk Menambah Kebutuhan Fungsional	
Kode Kebutuhan	RPBN-F-01
Objective	Analis dapat menambah kebutuhan fungsional.
Actor	Analis
Pre-condition	Halaman daftar kebutuhan fungsional sudah tersedia.
Main flow	1. Sistem menampilkan halaman daftar kebutuhan fungsional yang dilengkapi dengan tombol untuk menambah kebutuhan fungsional. 2. Aktor memilih tombol "Tambah kebutuhan fungsional". 3. Sistem menampilkan *form* untuk menambahkan kebutuhan fungsional yang terdiri dari kolom isian kode kebutuhan, deskripsi kebutuhan, estimasi waktu implementasi kebutuhan, dan estimasi biaya implementasi kebutuhan. Pada *form* juga tersedia tombol tambah dan batal. 4. Aktor mengisi kolom isian deskripsi kebutuhan, spesifikasi kebutuhan, estimasi waktu implementasi kebutuhan, dan estimasi biaya implementasi kebutuhan. 5. Aktor memilih tombol "Tambah". 6. Sistem melakukan validasi *form* dan menyimpan data kebutuhan fungsional.
Alternative flows	5. 1 Apabila aktor memilih tombol batal, maka kebutuhan tidak ditambahkan. 6.1 Apablia ada kolom isian yang kosong, sistem menampilkan pesan peringatan.
Post-condition	Kebutuhan fungsional tersimpan dalam *database* dan ditampilkan pada daftar kebutuhan fungsional.

Tabel 4.8 *Use Case Scenario* untuk Menambah Kebutuhan Non-Fungsional

Flow of Events untuk Menambah Kebutuhan Non-Fungsional

Kode Kebutuhan	RPBN-F-02
Objective	Analis dapat menambah kebutuhan non-fungsional.
Actor	Analis
Pre-condition	Halaman daftar kebutuhan non-fungsional sudah tersedia.
Main flow	1. Sistem menampilkan halaman daftar kebutuhan non-fungsional yang dilengkapi dengan tombol untuk menambah kebutuhan non-fungsional. 2. Aktor memilih tombol "Tambah kebutuhan non-fugsional". 3. Sistem menampilkan *form* untuk menambahkan kebutuhan non-fungsional yang terdiri dari kolom isian kode kebutuhan, dan deskripsi kebutuhan. Pada *form* juga tersedia tombol tambah dan batal. 4. Aktor mengisi kolom isian deskripsi kebutuhan pada *form*. 5. Aktor memilih tombol "Tambah". 6. Sistem melakukan validasi *form* dan menyimpan kebutuhan non-fungsional.
Alternative flows	5. 1 Apabila aktor memilih tombol batal, maka kebutuhan tidak ditambahkan. 6.1 Apabila ada kolom isian yang kosong, sistem menampilkan pesan untuk mengisi kolom tersebut.
Post-condition	Kebutuhan non-fungsional tersimpan dalam *database* dan ditampilkan pada daftar kebutuhan non-fungsional.

Tabel 4.9 *Use case scenario* untuk Melihat Daftar Kebutuhan Fungsional

Flow of Events untuk Melihat Daftar Kebutuhan Fungsional	
Kode Kebutuhan	RPBN-F-03
Objective	Analis dapat melihat daftar kebutuhan fungsional.
Actor	Analis
Pre-condition	Halaman utama sistem sudah tersedia.
Main flow	1. Aktor mengakses halaman daftar kebutuhan fungsional yang dilengkapi dengan tombol untuk menambah kebutuhan fungsional. 2. Sistem menampilkan daftar kebutuhan fungsional berdasarkan urutan prioritas tertinggi ke prioritas

	terendah. Data yang ditampilkan adalah kode kebutuhan, deskripsi kebutuhan, estimasi waktu, estimasi biaya, dan nilai prioritas. Pada setiap baris kebutuhan fungsional, tersedia tombol ubah dan hapus.
Alternative flows	1.1 Perluasan ke *use case* dengan kode kebutuhan RPBN-F-01. 2.1 Apabila belum dilakukan perhitungan prioritas kebutuhan, nilai prioritas bernilai 0. 2.2 Perluasan ke *use case* dengan kode kebutuhan RPBN-F-05, dan RPBN-F-07.
Post-condition	Daftar kebutuhan fungsional ditampilkan oleh sistem.

Tabel 4.10 *Use case scenario* untuk Melihat Daftar Kebutuhan Non-Fungsional

Flow of Events untuk Melihat Daftar Kebutuhan Non-Fungsional	
Kode Kebutuhan	RPBN-F-04
Objective	Analis dapat melihat daftar kebutuhan non-fungsional.
Actor	Analis
Pre-condition	Halaman utama sistem sudah tersedia.
Main flow	1. Aktor mengakses halaman daftar kebutuhan non-fungsional yang dilengkapi dengan tombol untuk menambah kebutuhan non-fungsional. 2. Sistem menampilkan daftar kebutuhan non-fungsional berdasarkan urutan prioritas tertinggi ke prioritas terendah. Data yang ditampilkan adalah kode kebutuhan, deskripsi kebutuhan, dan nilai prioritas. Pada setiap baris kebutuhan non-fungsional tersedia tombol ubah dan hapus.
Alternative flows	1.1 Perluasan ke *use case* dengan kode kebutuhan RPBN-F-02. 2.1 Apabila belum dilakukan perhitungan prioritas kebutuhan, nilai prioritas bernilai 0. 2.2 Perluasan ke *use case* dengan kode kebutuhan RPBN-RPBN-F-06, dan RPBN-F-07.
Post-condition	Daftar kebutuhan non-fungsional ditampilkan oleh sistem.

Tabel 4.11 *Use case scenario* untuk Mengubah Kebutuhan Fungsional

Flow of Events untuk Mengubah Kebutuhan Fungsional	
Kode Kebutuhan	RPBN-F-05
Objective	Analis dapat mengubah detail kebutuhan fungsional.
Actor	Analis
Pre-condition	Halaman daftar kebutuhan fungsional sudah tersedia.
Main flow	1. Sistem menampilkan daftar kebutuhan fungsional berdasarkan urutan prioritas tertinggi ke prioritas terendah. Data yang ditampilkan adalah kode kebutuhan, deskripsi kebutuhan, estimasi waktu, estimasi biaya, dan nilai prioritas. Pada setiap baris kebutuhan fungsional, tersedia tombol ubah dan hapus. 2. Aktor memilih tombol untuk mengubah kebutuhan. 3. Sistem menampilkan *form* perubahan kebutuhan yang terdiri dari kolom isian kode kebutuhan, deskripsi kebutuhan, estimasi waktu implementasi kebutuhan, dan estimasi biaya implementasi kebutuhan. Pada *form* tersedia tombol simpan dan batal. 4. Aktor melakukan perubahan pada kolom yang ingin diubah. 5. Aktor memilih tombol "Simpan". 6. Sistem melakukan validasi *form* dan menyimpan perubahan yang dibuat.
Alternative flows	5. 1 Apabila aktor memilih tombol "Batal" maka perubahan tidak dilakukan. 6.1 Apabila ada kolom isian yang kosong, maka sistem menampilkan pesan peringatan.
Post-condition	Perubahan pada kebutuhan fungsional tersimpan dalam *database* dan ditampilkan pada daftar kebutuhan.

Tabel 4.12 *Use case scenario* untuk Mengubah Kebutuhan Non-Fungsional

Flow of Events untuk Mengubah Kebutuhan Non-fungsional	
Kode Kebutuhan	RPBN-F-06
Objective	Analis dapat mengubah detail kebutuhan non-fungsional.
Actor	Analis

Pre-condition	Halaman daftar kebutuhan non-fungsional sudah tersedia.
Main flow	1. Sistem menampilkan daftar kebutuhan non-fungsional berdasarkan urutan prioritas tertinggi ke prioritas terendah. Data yang ditampilkan adalah kode kebutuhan, deskripsi kebutuhan, dan nilai prioritas. Pada setiap baris kebutuhan non-fungsional tersedia tombol ubah dan hapus. 2. Aktor memilih tombol untuk mengubah kebutuhan non-fungsional. 3. Sistem menampilkan *form* perubahan kebutuhan non-fungsional yang terdiri dari kolom isian kode kebutuhan dan deskripsi kebutuhan. Pada *form* tersedia tombol simpan dan batal. 4. Aktor melakukan perubahan pada kolom yang ingin diubah. 5. Aktor memilih tombol "Simpan". 6. Sistem melakukan validasi *form* dan menyimpan perubahan yang dibuat.
Alternative flows	5. 1 Apabila aktor memilih tombol "Batal" maka perubahan tidak dilakukan. 6.1 Apabila ada kolom isian yang kosong, maka sistem menampilkan pesan peringatan.
Post-condition	Perubahan pada kebutuhan non-fungsional tersimpan dalam *database* dan ditampilkan pada daftar kebutuhan.

Tabel 4.13 *Use case scenario* untuk Menghapus Kebutuhan Fungsional

Flow of Events untuk Menghapus Kebutuhan Fungsional	
Kode Kebutuhan	RPBN-F-07
Objective	Analis dapat menghapus kebutuhan fungsional.
Actor	Analis
Pre-condition	Halaman daftar kebutuhan fungsional sudah tersedia.
Main flow	1. Sistem menampilkan daftar kebutuhan fungsional berdasarkan urutan prioritas tertinggi ke prioritas terendah. Data yang ditampilkan adalah kode kebutuhan, deskripsi kebutuhan, estimasi waktu, estimasi biaya, dan nilai prioritas. Pada setiap baris kebutuhan fungsional, tersedia tombol ubah dan hapus.

	2. Aktor memlih tombol hapus pada kebutuhan yang ingin dihapus. 3. Sistem menampilkan pesan konfirmasi penghapusan yang dilengkapi tombol ya dan tidak. 4. Aktor memilih "Ya". 5. Sistem menghapus kebutuhan yang dipilih.
Alternative flows	4. 1 Apabila aktor memilih "Tidak" pada pesan konfirmasi penghapusan, maka kebutuhan tidak dihapus.
Post-condition	Kebutuhan yang dihapus tidak ditampilkan lagi pada daftar kebutuhan dan terhapus dari *database*.

Tabel 4.14 *Use case scenario* untuk Menghapus Kebutuhan Non-Fungsional

Flow of Events untuk Menghapus Kebutuhan Non-Fungsional	
Kode Kebutuhan	RPBN-F-08
Objective	Analis dapat menghapus kebutuhan non-fungsional.
Actor	Analis
Pre-condition	Halaman daftar kebutuhan non-fungsional tersedia.
Main flow	1. Sistem menampilkan daftar kebutuhan non-fungsional berdasarkan urutan prioritas tertinggi ke prioritas terendah. Data yang ditampilkan adalah kode kebutuhan, deskripsi kebutuhan, dan nilai prioritas. Pada setiap baris kebutuhan non-fungsional tersedia tombol ubah dan hapus. 2. Aktor memlih tombol hapus pada kebutuhan yang ingin dihapus. 3. Sistem menampilkan pesan konfirmasi penghapusan yang dilengkapi tombol ya dan tidak. 4. Aktor memilih "Ya". 5. Sistem menghapus kebutuhan yang dipilih.
Alternative flows	4. 1 Apabila aktor memilih "Tidak" pada pesan konfirmasi penghapusan, maka kebutuhan tidak dihapus.
Post-condition	Kebutuhan yang dihapus tidak ditampilkan lagi pada daftar kebutuhan dan terhapus dari *database*.

Tabel 4.15 *Use case scenario* untuk Menghitung Prioritas Kebutuhan

Flow of Events untuk Menghitung Prioritas Kebutuhan.	
Kode Kebutuhan	RPBN-F-09
Objective	Analis dapat melakukan perhitungan prioritas kebutuhan.
Actor	Analis.
Pre-condition	Halaman utama sistem sudah tersedia.
Main flow	1. Aktor mengakses halaman prioritasi kebutuhan. 2. Sistem menampilkan halaman penentuan prioritas kebutuhan non-fungsional yang di dalamnya terdapat informasi skala yang digunakan dan *form* yang terdiri dari kode serta deskripsi kebutuhan yang akan dibandingkan, kolom isian nilai kepentingan, dan *check box* dominan. 3. Aktor mengisi kolom isian nilai kepentingan dan menentukan kebutuhan yang dianggap lebih dominan. 4. Aktor memilih tombol untuk memproses perhitungan prioritas. 5. Sistem melakukan validasi *form*. 6. Sistem memproses perhitungan prioritas kebutuhan non-fungsional. 7. Sistem menampilkan hasil perhitungan prioritas kebutuhan non-fungsional dari prioritas tertinggi hingga terendah. Data yang ditampilkan adalah kode kebutuhan, deskripsi kebutuhan, dan nilai prioritas kebutuhan. 8. Aktor memilih tombol “Lanjutkan”. 9. Sistem menampilkan informasi skala yang digunakan dan *form* yang terdiri dari kode serta deskripsi kebutuhan yang akan dibandingkan dan kolom isian nilai kepentingan. 10. Aktor mengisi kolom isian nilai kepentingan. 11. Aktor memilih tombol “Selesai”. 12. Sistem melakukan validasi *form*. 13. Sistem memproses perhitungan prioritas kebutuhan fungsional. 14. Sistem menampilkan hasil perhitungan prioritas kebutuhan secara berurutan dari prioritas tertinggi

	hingga terendah. Data yang ditampilkan pada hasil perhitungan prioritas kebutuhan fungsional adalah kode kebutuhan, deskripsi kebutuhan, waktu, biaya, dan nilai prioritas kebutuhan.
Alternative flows	1. 1 Apabila belum ada kebutuhan non-fungsional dalam *database*, maka sistem menampilkan pesan bahwa kebutuhan non-fungsional belum ada dan menyediakan sarana untuk mengakses halaman kebutuhan non-fungsional. 5. 1 Apabila aktor mengisi kolom kepentingan dengan angka yang lebih kecil dari 1, maka sistem menampilkan pesan untuk mengisi kolom dengan angka yang lebih besar atau sama dengan 1. 5. 2 Apabila aktor mengisi kolom kepentingan dengan angka yang lebih besar dari 9, maka sistem menampilkan pesan untuk mengisi kolom dengan angka yang lebih kecil atau sama dengan 9. 5. 3 Apabila ada kolom nilai kepentingan yang belum diisi, sistem menampilkan pesan peringatan. 8. 1 Apabila belum ada kebutuhan fungsional dalam *database*, maka sistem menampilkan pesan bahwa kebutuhan fungsional belum ada dan menyediakan sarana untuk mengakses halaman kebutuhan fungsional. 12. 1 Apabila aktor mengisi kolom kepentingan dengan angka yang lebih kecil dari 1, maka sistem menampilkan pesan untuk mengisi kolom dengan angka yang lebih besar atau sama dengan 1. 12. 2 Apabila aktor mengisi kolom kepentingan dengan angka yang lebih besar dari 5, maka sistem menampilkan pesan untuk mengisi kolom dengan angka yang lebih kecil atau sama dengan 5. 12. 3 Apabila ada kolom nilai kepentingan yang belum diisi, sistem menampilkan pesan peringatan.
Post-condition	Didapatkan nilai prioritas dari setiap kebutuhan.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan

Perancangan merupakan tahapan yang dilakukan setelah semua kebutuhan sistem didapatkan dari tahapan analisis kebutuhan. Perancangan sistem dalam pengembangan aplikasi penentuan prioritas kebutuhan ini dilakukan pada empat atribut utama sistem, yaitu perancangan arsitektur, perancangan komponen, perancangan data, dan perancangan antarmuka.

5.1.1 Perancangan Arsitektur

Perancangan arsitektur dilakukan dengan menggunakan *sequence diagram* dan *class diagram*. *Sequence diagram* menggambarkan secara rinci alur pertukaran pesan antar objek di dalam sistem, sedangan *class diagram* menggambarkan hubungan antar klas yang menyusun sistem.

5.1.1.1 *Sequence Diagram*

Bagian ini menjelaskan tiga sampel *sequence diagram* dari sistem penentuan prioritas kebutuhan fungsional berdasarkan kebutuhan non-fungsional, yaitu *sequence diagram* untuk menambah kebutuhan fungsional, menghitung prioritas kebutuhan, dan mengubah kebutuhan non-fungsional.

5.1.1.1.1 *Sequence Diagram* Menambah Kebutuhan Fungsional

Sequence diagram untuk menambah kebutuhan fungsional dapat dilihat pada Gambar 5.1. Aktor yang terlibat dalam *sequence diagram* menambah kebutuhan fungsional adalah analis. Objek-objek yang terdapat dalam *sequence diagram* ini adalah satu objek *boundary*, yaitu “kebutuhanf”, satu objek *controller*, yaitu “AnalisController”, dan dua objek *entities*, yaitu “DaftarKebutuhanFungsional” dan “KebutuhanFungsional”.

5.1.1.1.2 *Sequence Diagram* Mengubah Kebutuhan Non-Fungsional

Sequence diagram untuk mengubah kebutuhan non-fungsional dapat dilihat pada Gambar 5.3. Objek-objek yang terdapat dalam *sequence diagram* ini adalah dua objek *boundaries*, yaitu “kebutuhannf” dan “editNF”, kemudian satu objek *controller*, yaitu “AnalisController” dan dua objek *entities*, yaitu “DaftarKebutuhanNonfungsional” dan “KebutuhanNonfungsional”.

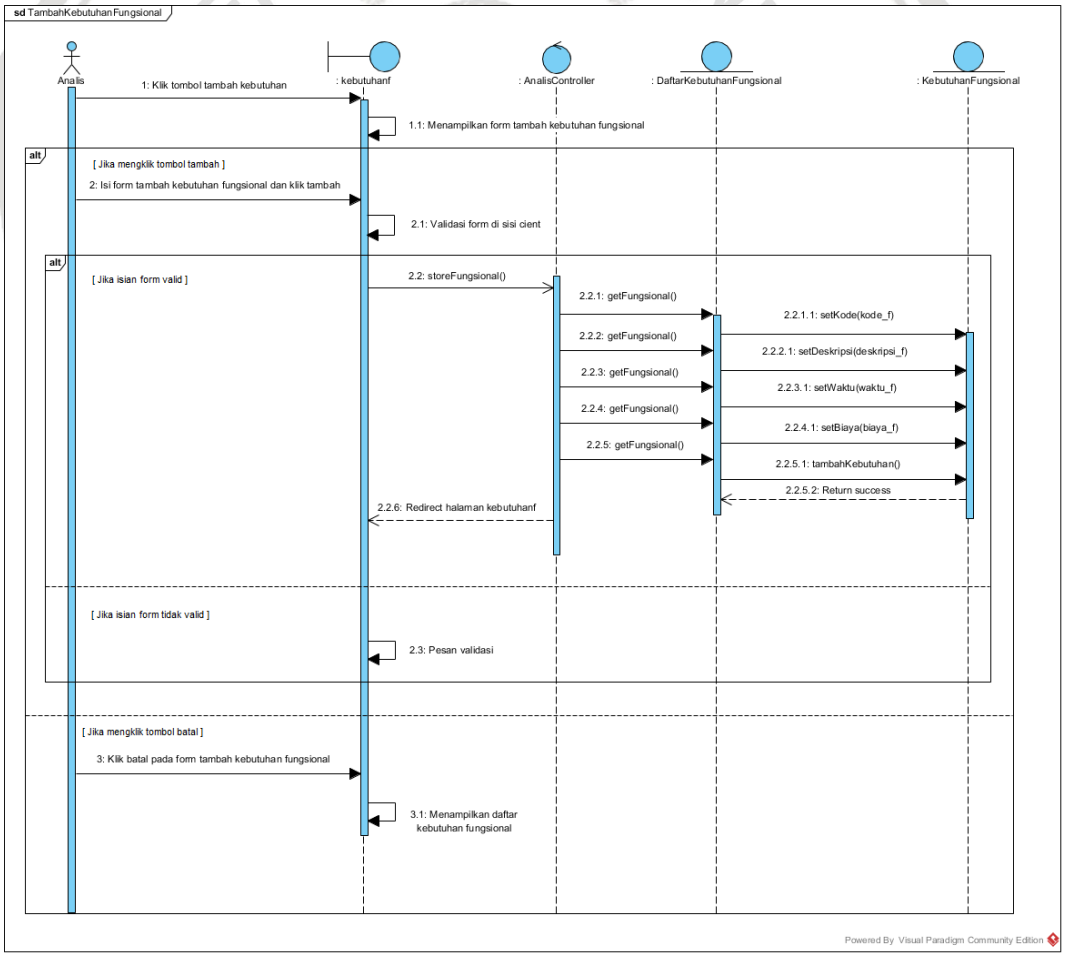
5.1.1.1.3 *Sequence Diagram* Menghitung Prioritas Kebutuhan

Sequence diagram untuk menghitung prioritas kebutuhan dapat dilihat pada Gambar 5.2. Objek-objek yang terdapat dalam *sequence diagram* menghitung prioritas kebutuhan adalah empat objek *boundaries*, yaitu “prioritasinf”, “prioritasif”, “hasilnf” dan “hasilf”, kemudian satu objek *controller*, yaitu “AnalisController” dan empat objek *entities,* yaitu “KebutuhanNonfungsional”, “KebutuhanFungsional”, “DaftarKebutuhanNonfungsional” dan “DaftarKebutuhanFungsional”.

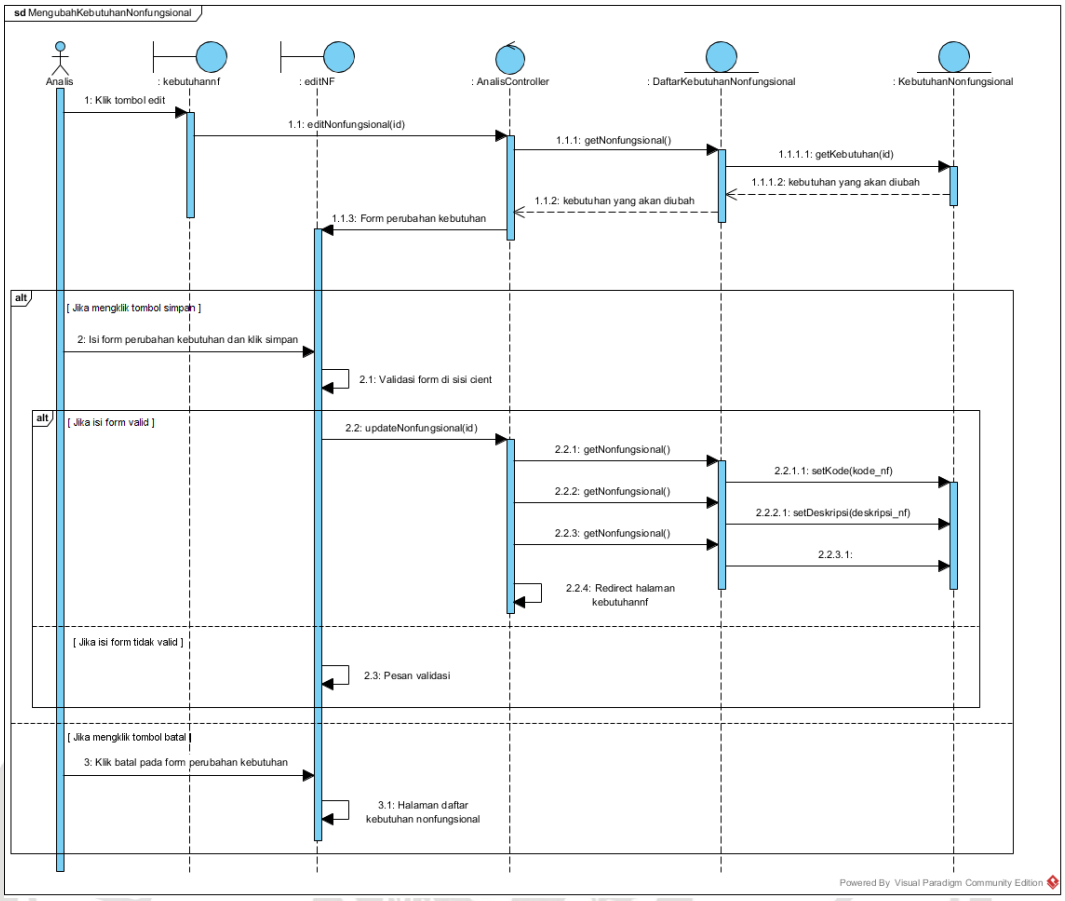
5.1.1.2 *Class Diagram*

Class diagram menjelaskan mengenai klas-klas yang membentuk aplikasi penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional beserta hubungan antar klas-klas tersebut dalam bentuk *class diagram*.

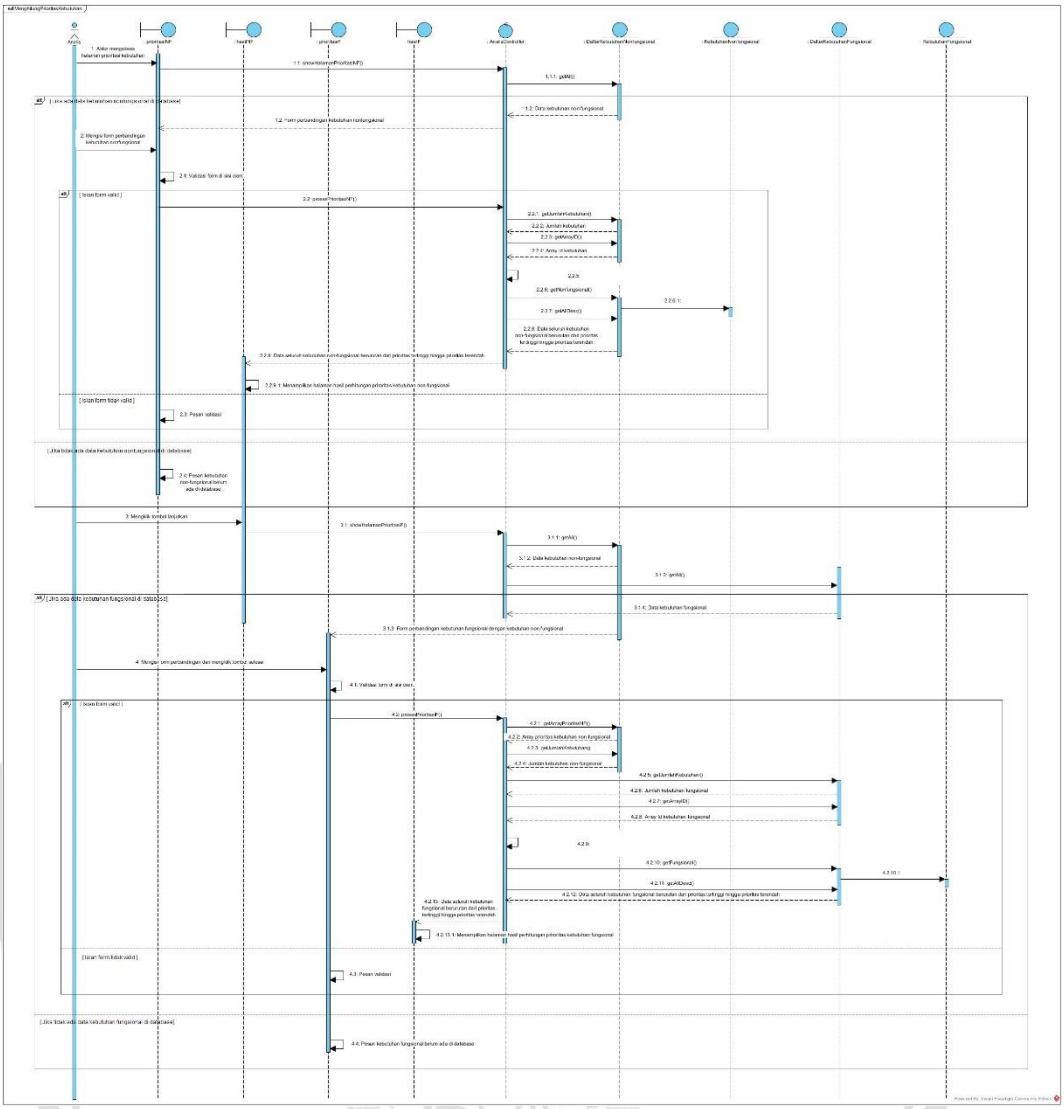
Pemodelan *class diagram* umum terdapat pada Gambar 5.4, detil pemodelan *class diagram controller* terdapat pada Gambar 5.5, dan detil pemodelan *class diagram* model terdapat pada Gambar 5.6. Secara garis besar terdapat dua jenis penyusun sistem penentuan prioritas ini, yaitu klas *controller* dan klas *model*. Klas CI_Controller adalah *general class* dari klas AnalisController. Klas CI_Model adalah *general class* dari klas Kebutuhan, DaftarKebutuhanFungsional, dan DaftarKebutuhanNonfungsional. Klas Kebutuhan merupakan *abstract class* yang dapat dispesialisasikan menjadi klas KebutuhanFungsional dan klas KebutuhanNonfungsional. Klas DaftarKebutuhanNonfungsional memiliki hubungan agregasi dengan klas KebutuhanNonfungsional" dan klas DaftarKebutuhanFungsional memiliki hubungan agregasi dengan klas KebutuhanFungsional.



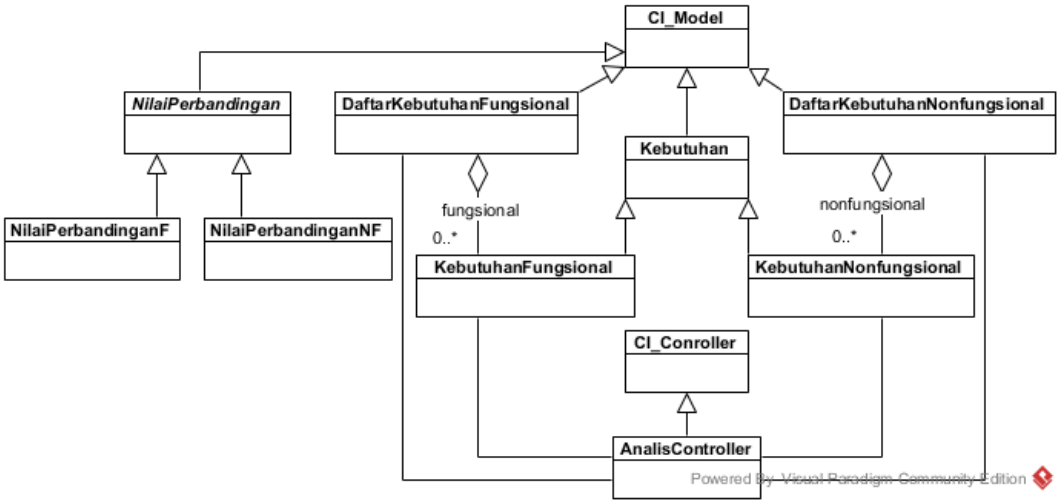
Gambar 5.1 *Sequence diagram* menambah kebutuhan fungsional



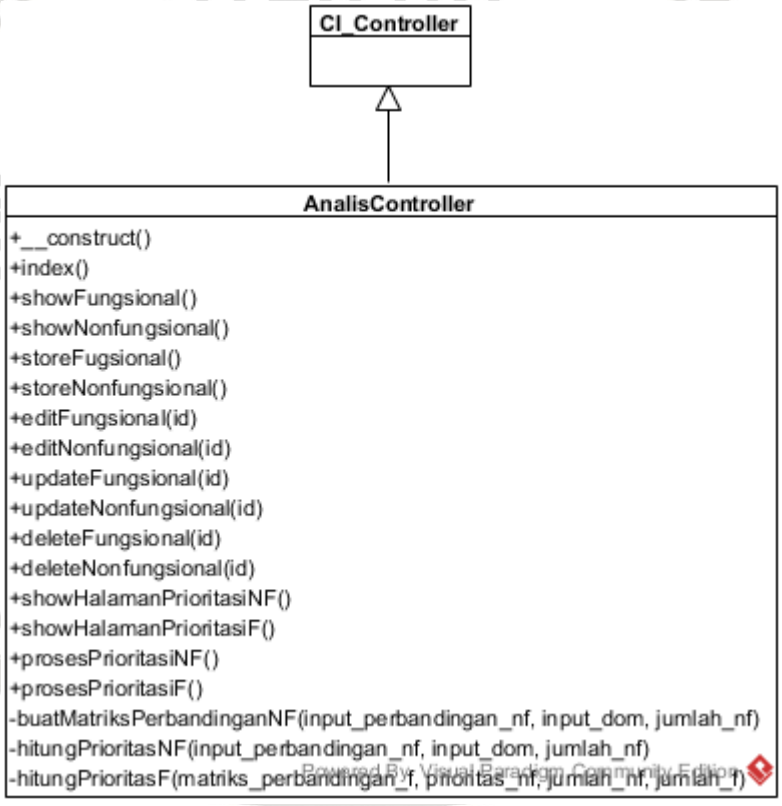
Gambar 5.2 *Sequence diagram* mengubah kebutuhan non-fungsional



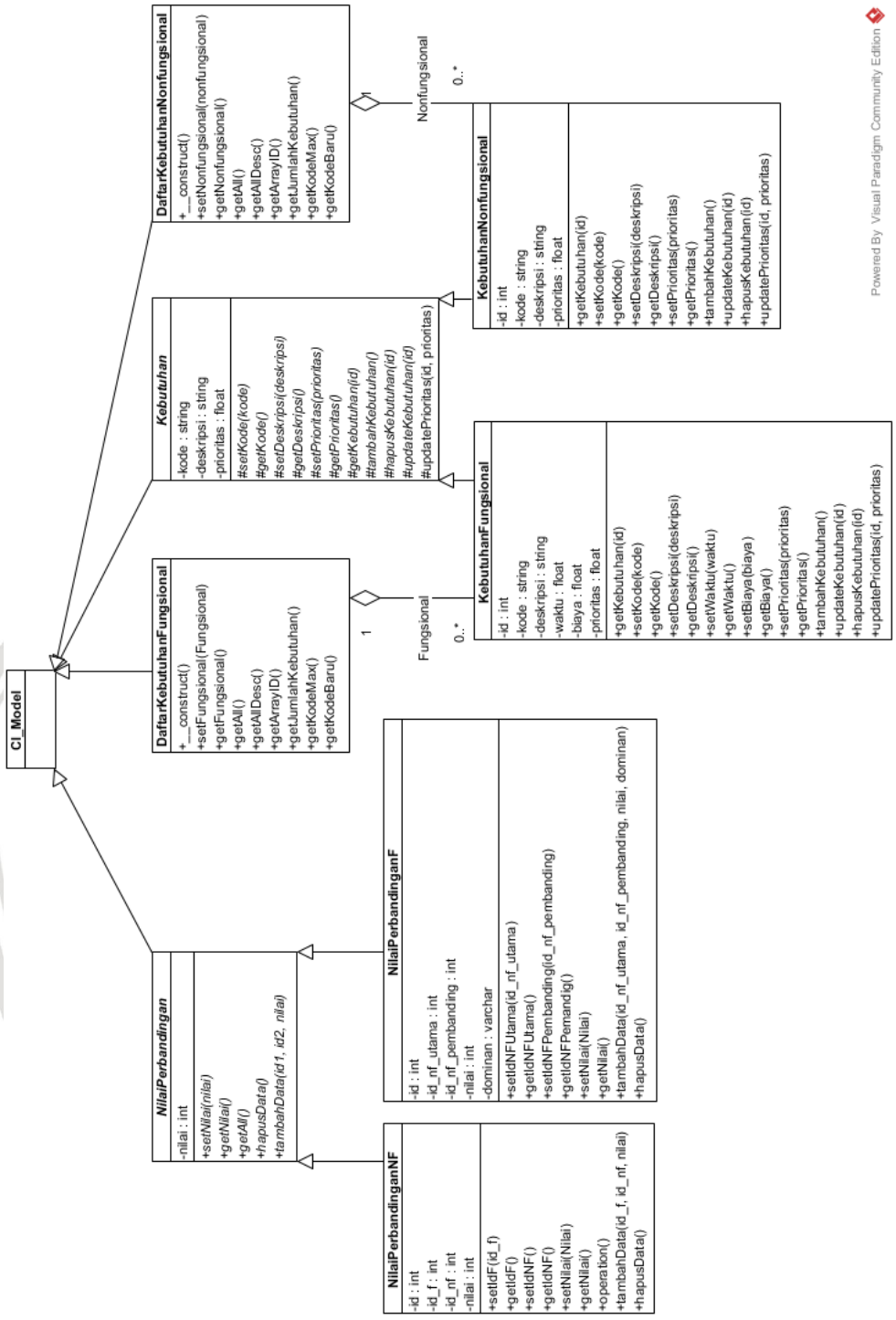
Gambar 5.3 *Sequence diagram* menghitung prioritas kebutuhan



Gambar 5.4 Pemodelan *Class Diagram* Umum



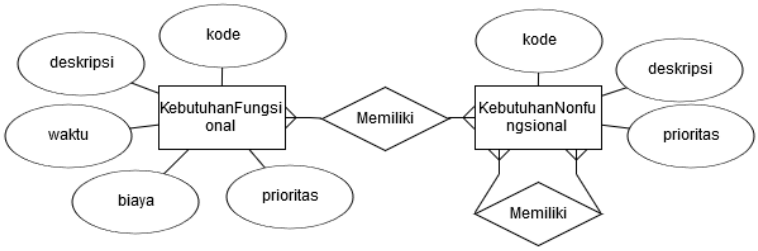
Gambar 5.5 Pemodelan *Class Diagram Controller*



Gambar 5.6 Pemodelan *Class Diagram Model*

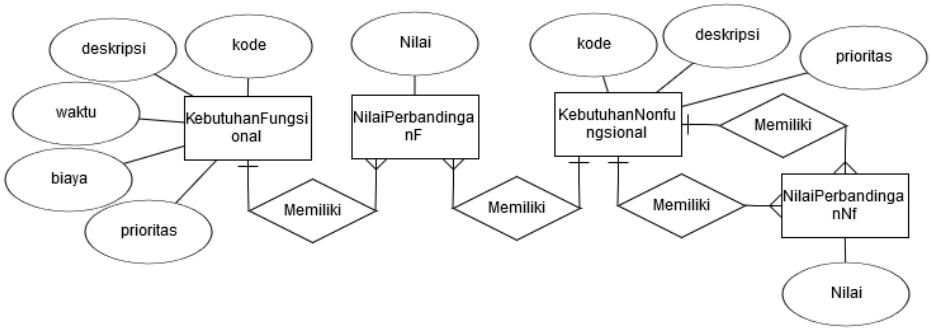
5.1.2 Perancangan Data

Perancangan data dilakukan dengan membuat *Conceptual Data Model* (CDM), dan *Physical Data Model* (PDM). CDM digunakan untuk memodelkan entitas yang ada pada sistem, atribut yang dimiliki entitas, dan relasi di antara entitas. CDM untuk sistem ini terdapat pada Gambar 5.7 dan Gambar 5.8.



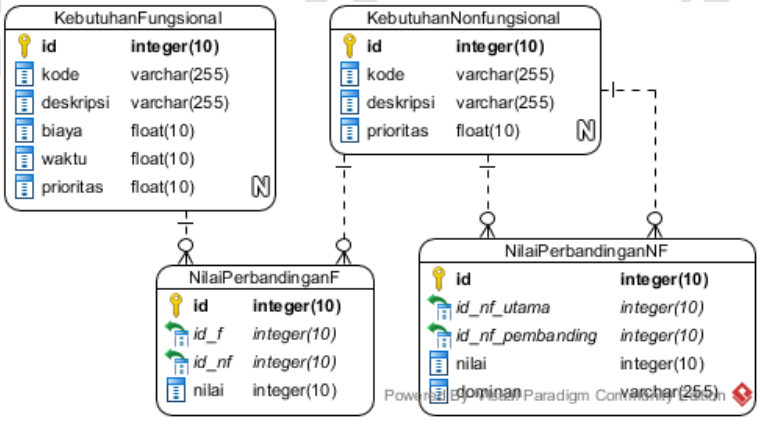
Gambar 5.7 *Conceptual Data Model*

CDM pada Gambar 5.7 menggambarkan relasi *many-to-many* antara entitas KebutuhanFungsional dan KebutuhanNonfungsional juga relasi *unary many-to-many* pada entitas KebutuhanNonfungsional. Dari relasi *many-to-many* tersebut dapat dibentuk dua tabel baru untuk menyimpan nilai yang dihasilkan dari relasi tersebut, yaitu entitas NilaiPerbandinganF dan NilaiPerbandinganNF. Gambar 5.8 berikut merupakan CDM dengan dua tabel tambahan.



Gambar 5.8 *Conceptual Data Model* dengan Dua Entitas Tambahan

CDM kemudian digunakan menjadi dasar dalam pemodelan dalam bentuk PDM. Dalam PDM, pemodelan lebih lengkap yaitu disertai *primary key*, *foreign keys*, nama tabel, nama kolom, dan juga tipe data dari setiap kolom. PDM terdapat pada Gambar 5.9.



Gambar 5.9 *Physical Data Model*

5.1.3 Perancangan Komponen

Perancangan komponen dilakukan dengan mendefinisikan algoritme dari proses yang terjadi dalam komponen perangkat lunak. Komponen dalam pendekatan berorientasi objek merupakan klas yang membangun sistem. Perancangan komponen yang dijelaskan disini adalah algoritme untuk *method* prosesPrioritasiNF dari klas analisController, *method* getAllDesc dari klas DaftarKebutuhanFungsional dan *method* updatePrioritas dari klas Kebutuhannonfungsional.

5.1.3.1 Perancangan Komponen Klas AnalisController

Nama *method:* prosesPrioritasiNF

Algoritme:

Tabel 5.1 *Pseudocode method* prosesPrioritasiNF

No	*Pseudocode*
1	Start
2	jumlah_nf = memanggil method getJumlahKebutuhan() dengan objek nonfungsional
3	id_nf = memanggil method getArrayID() dengan objek nonfungsional
4	input_perbandingan_nf = input post 'kepentingan'
5	input_dom = input post 'dom'
6	id_nf_utama = imput post 'id_utama'
7	id_nf_pembanding = imput post 'id_pembanding'
8	np = merge array input_perbandingan_nf
9	dom = merge array input_dom
10	Memanggil method simpanPerbadinganNF (id_nf_utama, id_nf_pembanding, np, dom)
11	prioritas_nf = hitungPrioritasNF(input_perbandingan_nf, input_dom, jumlah_nf)
12	for x=0 to jumlah_nf-1
13	memanggil method updatePrioritas(id_nf[x], prioritas_nf[x]) dengan objek nonfungsional
14	endfor
15	nf = memanggil method getAllDesc dengan objek nonfungsional
16	Menampilkan halaman hasilPrioritasiNF
17	End

5.1.3.2 Perancangan Komponen Klas DaftarKebutuhanFungsional

Nama *method:* getAllDesc

Algoritme:

Tabel 5.2 *Pseudocode method* getAllDesc

No	*Pseudocode*
1	Start

2	Query mengurutkan data kebutuhan non-fungsional berdasarkan nilai prioritas secara *descending*
3	Query mengambil seluruh data kebutuhan non-fungsional yang sudah berurutan
4	End

5.1.3.3 Perancangan Komponen Klas Kebutuhannonfungsional

Nama *method:* updatePrioritas

Algoritme:

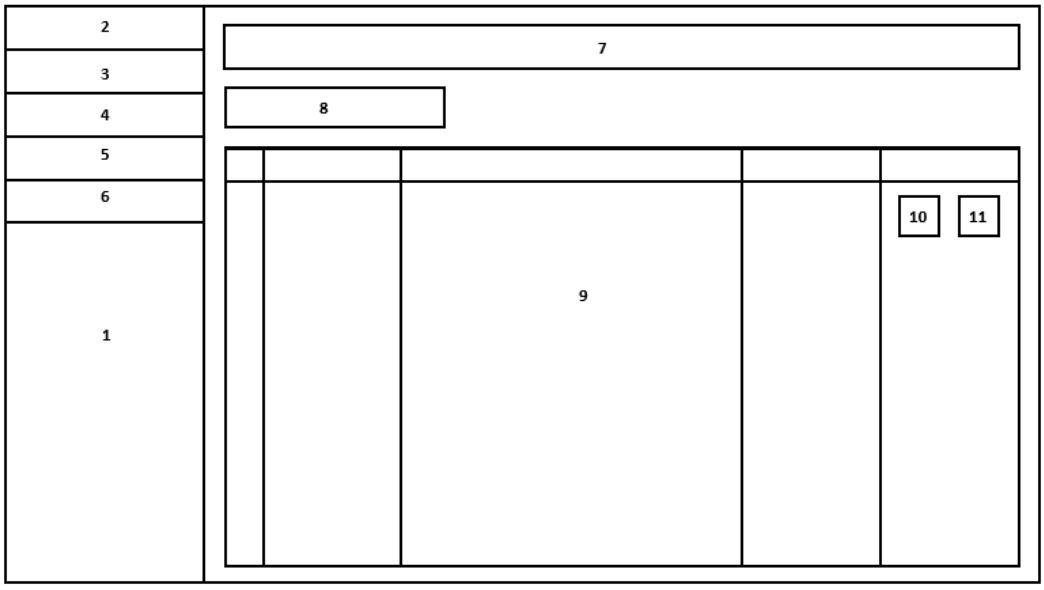
Tabel 5.3 *Pseudocode method* updatePrioritas

No	*Pseudocode*
1	Start
2	inisialisasi variabel data = prioritas
3	Query find data kebutuhan non-fungsional dengan id = id
4	Query update nilai prioritas kebutuhan non-fungsional dengan nilai pada variabel data
5	End

5.1.4 Perancangan Antarmuka

5.1.4.1 Perancangan Antarmuka Halaman Daftar Kebutuhan Fungsional

Gambar 5.10 merupakan rancangan dari antarmuka halaman Daftar Kebutuhan Fungsional. Dan penjelasan dari rancangan tersebut terdapat pada tabel 5.4.



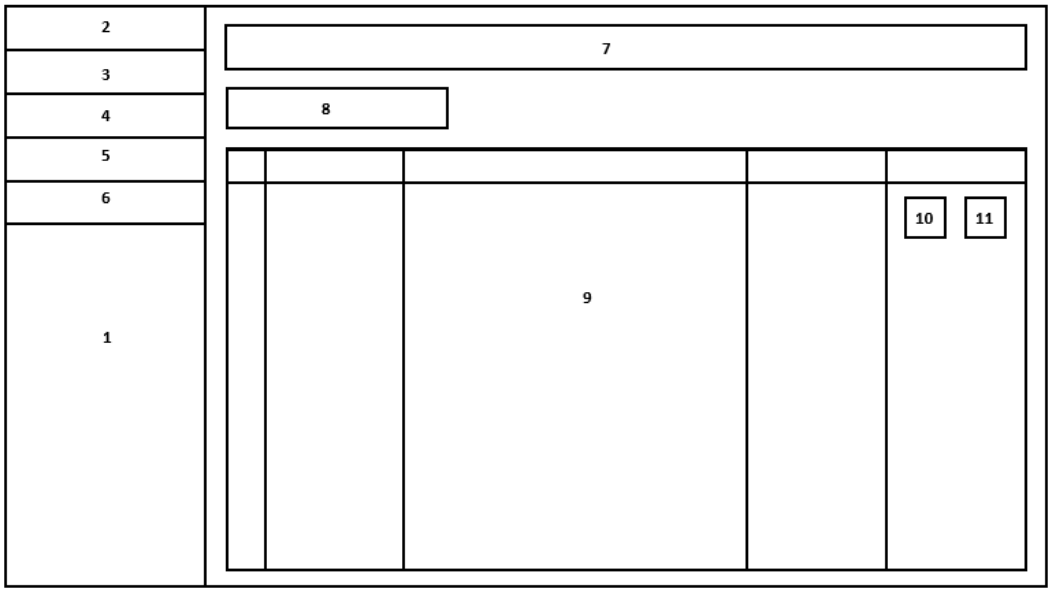
Gambar 5.10 Perancangan Antarmuka Halaman Daftar Kebutuhan Fungsional

Tabel 5.4 Penjelasan antarmuka halaman Daftar Kebutuhan Fungsional

No	Nama Objek	Tipe	Keterangan
1	*Sidebar menu*	*Sidebar*	*Sidebar* yang berisi menu aplikasi.
2	Pilihan menu Tutorial	Menu	Menu untuk masuk halaman Tutorial.
3	Pilihan menu Daftar Kebutuhan	Menu	Menu *disabled* Daftar Kebutuhan.
4	Pilihan menu Kebutuhan Fungsional	Menu	Menu untuk masuk halaman daftar kebutuhan fungsional.
5	Pilihan menu Kebutuhan Non-Fungsional	Menu	Menu untuk masuk halaman daftar kebutuhan non-fungsional.
6	Pilihan menu Prioritasi Kebutuhan	Menu	Menu untuk masuk halaman prioritasi kebutuhan.
7	Judul Kebutuhan	*Text*	Judul Kebutuhan Fungsional.
8	Tombol tambah kebutuhan fungsional	Tombol	Tombol untuk membuka *modal form* untuk menambahkan kebutuhan fungsional.
9	Tabel kebutuhan fungsional	Tabel	Tabel yang berisi daftar seluruh kebutuhan fungsional.
10	Tombol Edit	Tombol	Tombol untuk mengedit kebutuhan fungsional.
11	Tombol Hapus	Tombol	Tombol untuk menghapus kebutuhan fungsional.

5.1.4.2 Perancangan Antarmuka Halaman Daftar Kebutuhan Non-Fungsional

Gambar 5.11 merupakan rancangan dari antarmuka halaman Daftar Kebutuhan Non-Fungsional. Dan penjelasan dari rancangan tersebut terdapat pada tabel 5.5.



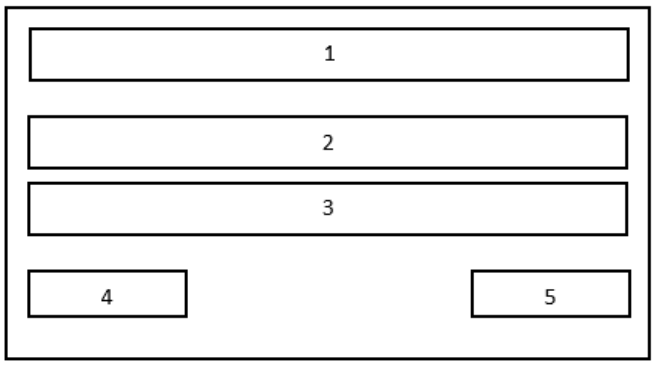
Gambar 5.11 Perancangan Antarmuka Halaman Daftar Kebutuhan Non-Fungsional

Tabel 5.5 Penjelasan antarmuka halaman Daftar Kebutuhan Non-Fungsional

No	Nama Objek	Tipe	Keterangan
1	*Sidebar menu*	*Sidebar*	*Sidebar* yang berisi menu aplikasi.
2	Pilihan menu Tutorial	Menu	Menu untuk masuk halaman Tutorial.
3	Pilihan menu Daftar Kebutuhan	Menu	Menu *disabled* Daftar Kebutuhan.
4	Pilihan menu Kebutuhan Fungsional	Menu	Menu untuk masuk halaman daftar kebutuhan fungsional.
5	Pilihan menu Kebutuhan Non-Fungsional	Menu	Menu untuk masuk halaman daftar kebutuhan non-fungsional.
6	Pilihan menu Prioritasi Kebutuhan	Menu	Menu untuk masuk halaman prioritasi kebutuhan.
7	Judul Kebutuhan	*Text*	Judul Kebutuhan Non-Fungsional.
8	Tombol tambah kebutuhan non-fungsional	Tombol	Tombol untuk membuka *modal form* untuk menambahkan kebutuhan non-fungsional.

9	Tabel kebutuhan non-fungsional	Tabel	Tabel yang berisi daftar seluruh kebutuhan non-fungsional.
10	Tombol Edit	Tombol	Tombol untuk mengedit kebutuhan non-fungsional.
11	Tombol Hapus	Tombol	Tombol untuk menghapus kebutuhan non-fungsional.

Gambar 5.12 berikut ini merupakan rancangan dari *modal form* tambah Kebutuhan Non-Fungsional. Dan penjelasan dari rancangan tersebut terdapat pada tabel 5.6.



Gambar 5.12 Perancangan Antarmuka *Modal Form* Tambah Kebutuhan Non-Fungsional

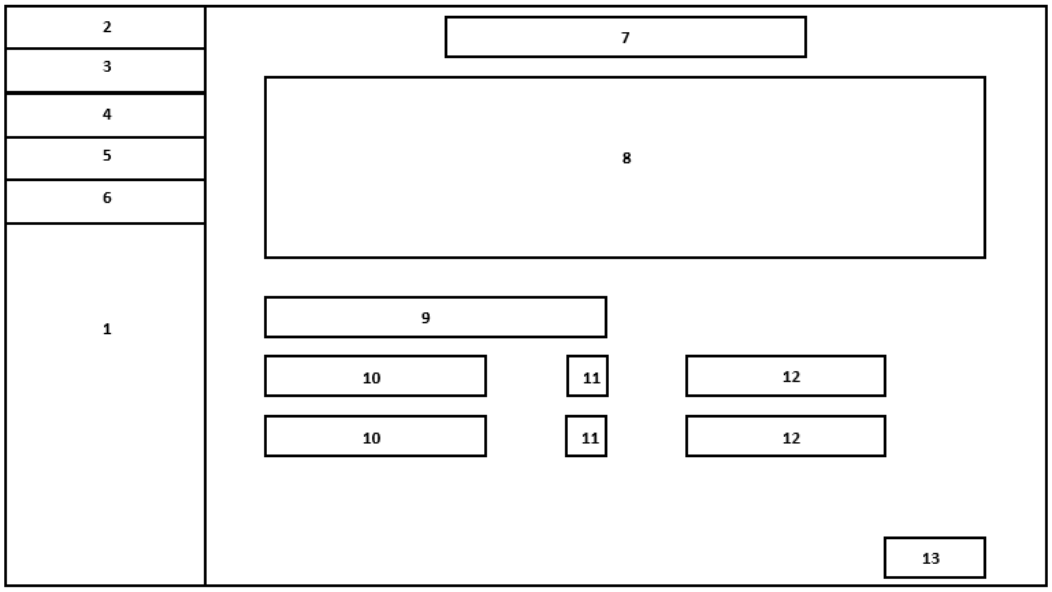
Tabel 5.6 Penjelasan antarmuka *modal form* tambah kebutuhan non-fungsional

No	Nama Objek	Tipe	Keterangan
1	Judul *form*	*Text*	Judul *form* Tambah Kebutuhan Non-Fungsional.
2	Kolom kode kebutuhan	*Textfield*	Kolom kode kebutuhan non-fungsional yang bersifat *readonly*.
3	Kolom deskripsi kebutuhan	*Textfield*	Kolom untuk mengisi deskripsi kebutuhan non-fugsional.
4	Tombol batal	Tombol	Tombol untuk membatalkan penambahan kebutuhan.

5	Tombol tambah	Tombol	Tombol untuk menyimpan kebutuhan yang baru ditambahkan.

5.1.4.3 Perancangan Antarmuka Halaman Prioritasi Kebutuhan Non-Fungsional

Gambar 5.13 merupakan rancangan dari antarmuka halaman Prioritasi Kebutuhan Non-Fungsional. Dan penjelasan dari rancangan tersebut terdapat pada tabel 5.7.



Gambar 5.13 Perancangan Antarmuka Halaman Prioritasi Kebutuhan Non-Fungsional

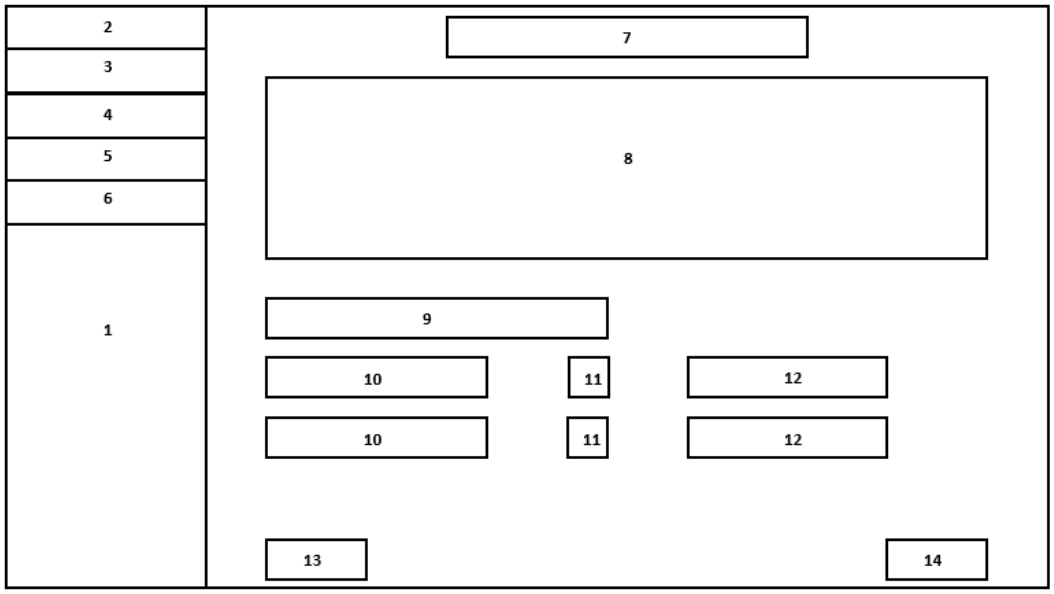
Tabel 5.7 Penjelasan antarmuka halaman Prioritasi Kebutuhan Non-Fungsional

No	Nama Objek	Tipe	Keterangan
1	*Sidebar menu*	*Sidebar*	*Sidebar* yang berisi menu aplikasi.
2	Pilihan menu Tutorial	Menu	Menu untuk masuk halaman Tutorial.
3	Pilihan menu Daftar Kebutuhan	Menu	Menu *disabled* Daftar Kebutuhan.
4	Pilihan menu Kebutuhan Fungsional	Menu	Menu untuk masuk halaman daftar kebutuhan fungsional.
5	Pilihan menu Kebutuhan Non-Fungsional	Menu	Menu untuk masuk halaman daftar kebutuhan non-fungsional.

6	Pilihan menu Prioritasi Kebutuhan	Menu	Menu untuk masuk halaman prioritasi kebutuhan.
7	Judul prioritasi	*Text*	Judul Prioritasi Non-Fungsional.
8	Keterangan skala	*Text*	Keterangan skala yang digunakan untuk prioritasi kebutuhan non-fungsional
9	Kode dan Deskripsi Kebutuhan Utama	*Text*	Kode kebutuhan serta deskripsi kebutuhan non-fungsional yang akan dibandingkan.
10	Kode dan Deskripsi Kebutuhan Pembanding	*Text*	Kode kebutuhan serta deskripsi kebutuhan non-fungsional yang menjadi pembanding.
11	*Checkbox* dominan	*Checkbox*	*Checkbox* yang digunakan untuk menentukan apakah kebutuhan pmabnding lebih dominan dari kebutuhan utama.
12	Nilai kepentingan	*Input area*	*Field* yang berupa *input area* untuk menentukan nilai kepentingan.
13	Tombol lanjutkan	Tombol	Tombol untuk melanjutkan proses prioritasi.

5.1.4.4 Perancangan Antarmuka Halaman Prioritasi Kebutuhan Fungsional

Gambar 5.14 berikut ini merupakan rancangan dari antarmuka halaman Prioritasi Kebutuhan Fungsional. Dan penjelasan dari rancangan tersebut terdapat pada tabel 5.8.



Gambar 5.14 Perancangan Antarmuka Halaman Prioritasi Kebutuhan Fungsional

Tabel 5.8 Penjelasan antarmuka halaman Prioritasi Kebutuhan Fungsional

No	Nama Objek	Tipe	Keterangan
1	*Sidebar menu*	*Sidebar*	*Sidebar* yang berisi menu aplikasi.
2	Pilihan menu Tutorial	Menu	Menu untuk masuk halaman Tutorial.
3	Pilihan menu Daftar Kebutuhan	Menu	Menu *disabled* Daftar Kebutuhan.
4	Pilihan menu Kebutuhan Fungsional	Menu	Menu untuk masuk halaman daftar kebutuhan fungsional.
5	Pilihan menu Kebutuhan Non-Fungsional	Menu	Menu untuk masuk halaman daftar kebutuhan non-fungsional.
6	Pilihan menu Prioritasi Kebutuhan	Menu	Menu untuk masuk halaman prioritasi kebutuhan.
7	Judul prioritasi	*Text*	Judul Prioritasi Non-Fungsional.
8	Keterangan skala	*Text*	Keterangan skala yang digunakan untuk prioritasi kebutuhan fungsional

9	Kode dan Deskripsi Kebutuhan Utama	*Text*	Kode kebutuhan serta deskripsi kebutuhan fungsional yang akan dibandingkan.
10	Kode dan Deskripsi Kebutuhan Pembanding	*Text*	Kode kebutuhan serta deskripsi kebutuhan non-fungsional yang menjadi pembanding.
11	*Checkbox* dominan	*Checkbox*	*Checkbox* yang digunakan untuk menentukan apakah kebutuhan pmabnding lebih dominan dari kebutuhan utama.
12	Nilai kepentingan	*Input area*	*Field* yang berupa *input area* untuk menentukan nilai kepentingan.
13	Tombol kembali	Tombol	Tombol untuk kembali ke halaman sebelumnya.
14	Tombol selesai	Tombol	Tombol untuk memproses prioritasi.

5.2 Implementasi

5.2.1 Spesifikasi Sistem

Tabel 5.9 merupakan spesifikasi dari perangkat keras yang digunakan.

Tabel 5.9 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Laptop	Asus A455L
RAM	4GB
Hard disk	1TB

Tabel 5.10 merupakan spesifikasi dari perangkat lunak yang digunakan dalam pengembangan aplikasi penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungisonal.

Tabel 5.10 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
Sistem Operasi	Windows 8

Editor Dokumentasi	Microsoft Word 2013
Editor Perancangan	Visual Paradigm, Enterprise Architect
Editor Pemrograman	Visual Studio Code
Bahasa Pemrograman	HTML, PHP, Javascript, CSS
Framework	CodeIgniter, MaterializeCSS
DBMS	Xampp MySql
Browser	Mozilla Firefox, Google Chrome

5.2.2 Implementasi Kode Program

5.2.2.1 Implementasi *Method* prosesPriroritasiNF *Class* analisController

Implementasi kode program untuk *method* prosesPrioritasiNF dari klas analisController terdapat pada Gambar 5.11.

Tabel 5.11 *Sourcecode method* prosesPrioritasiNF

No	*Sourcecode*
1	`public function prosesPrioritasiNF()`
2	`{`
3	`    $jumlah_nf = $this->daftarNonfungsional->getJumlahKebutuhan();`
4	`    $id_nf = $this->daftarNonfungsional->getArrayID();`
5	
6	`    $input_perbandingan_nf = $this->input->post('kepentingan');`
7	`    $input_dom = $this->input->post('dom');`
8	
9	`    $id_nf_utama = $this->input->post('id_nf_utama');`
10	`    $id_nf_pembanding = $this->input->post('id_nf_pembanding');`
11	
12	`    $np = array_reduce($input_perbandingan_nf, 'array_merge', array());`
13	`    $dom = array_reduce($input_dom, 'array_merge', array());`
14	
15	`    $this->simpanPerbandinganNF($id_nf_utama, $id_nf_pembanding, $np, $dom);`
16	
17	`    $prioritas_nf = $this->hitungPrioritasNF($input_perbandingan_nf, $input_dom, $jumlah_nf);`
18	
19	`    //untuk update nilai prioritas pada database`
20	`    for($x=0; $x<$jumlah_nf; $x++){`
21	`        $this->daftarNonfungsional->getNonfungsional()->updatePrioritas($id_nf[$x], $prioritas_nf[$x]);`
22	`    }`
23	
24	`    // load view untuk menampilkan hasil prioritas`
25	`    $nf['nonfungsionals'] = $this->daftarNonfungsional->getAllDesc();`
26	`    $this->load->view('elements/headerMtr');`
27	`    $this->load->view('prioritasi/hasilnfMtr', $nf);`
28	`    $this->load->view('elements/footerMtr');`

29	}

5.2.2.2 Implementasi *Method* getAllDesc *Class* DaftarKebutuhanFungsional

Implementasi kode program untuk *method* getAllDesc dari klas *DaftarKebutuhanFungsional* terdapat pada Gambar 5.12.

Tabel 5.12 *Sourcecode method* getAllDesc

No	*Sourcecode*
1	`public function getAllDesc(){`
2	`    $this->db->order_by('prioritas', 'DESC');`
3	`    return $this->db->get('fungsionals');`
4	`}`

5.2.2.3 Implementasi *Method* updatePrioritas *Class* "Kebutuhannonfungsional"

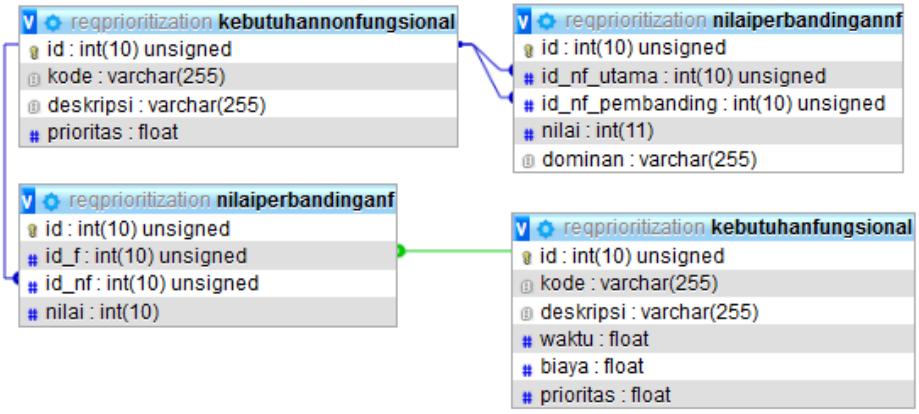
Implementasi kode program untuk *method* updatePrioritas dari klas *Kebutuhannonfungsional* terdapat pada Gambar 5.13.

Tabel 5.13 *Sourcecode method* updatePrioritas

No	*Sourcecode*
1	`public function updatePrioritas($id, $prioritas){`
2	`    $data = array('prioritas'=> $prioritas);`
3	`    $this->db->where('id', $id);`
4	`    $this->db->update('nonfungsionals', $data);`
5	`}`

5.2.3 Implementasi Data

Implementasi data dibuat berdasarkan perancangan data yang telah dibuat. Gambar 5.15 menggambarkan semua tabel dalam basis data dalam sistem beserta kolom dan tipe data untuk masing-masing kolomnya.



Gambar 5.15 Implementasi Data

5.2.4 Implementasi Antarmuka

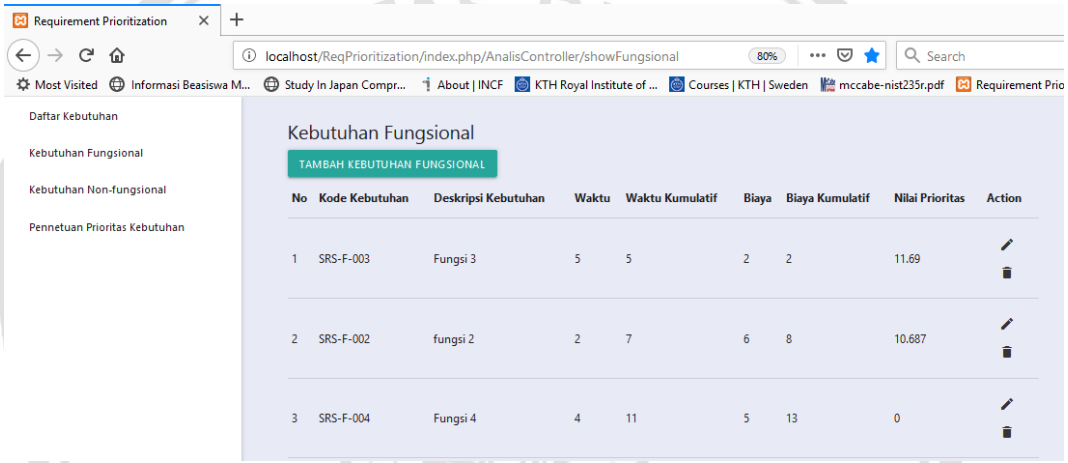
Pada bagian ini ditampilkan hasil implementasi antarmuka dari perancangan yang sebelumnya sudah dilakukan.

5.2.4.1 Implementasi Antarmuka Halaman Daftar Kebutuhan Fungsional

Gambar 5.16 berikut ini merupakan hasil implementasi antarmuka halaman Daftar Kebutuhan Fungsional. Gambar 5.17 merupakan hasil implementasi antarmuka *modal form* Tambah Kebutuhan Fungsional.

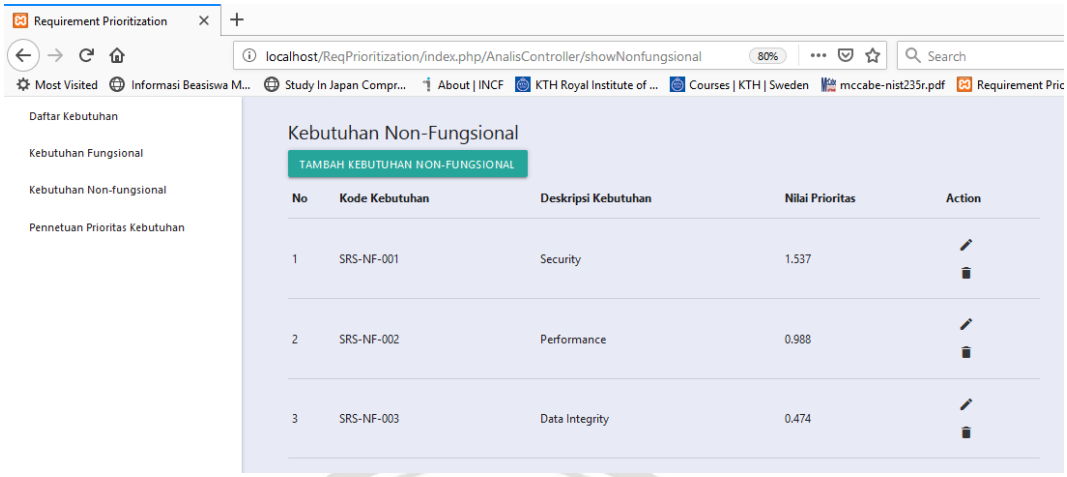
5.2.4.2 Implementasi Antarmuka Halaman Daftar Kebutuhan Fungsional

Gambar 5.18 merupakan hasil implementasi antarmuka halaman Daftar Kebutuhan Fungsional. Gambar 5.19 merupakan hasil implementasi antarmuka *modal form* Tambah Kebutuhan Fungsional.

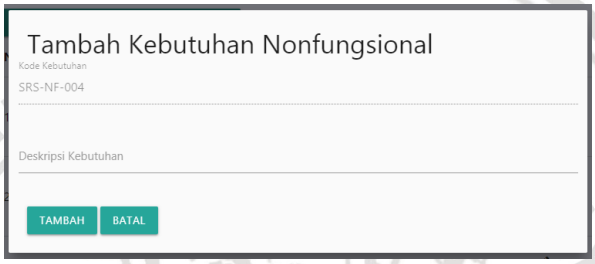


Gambar 5.16 Implementasi antarmuka halaman Daftar Kebutuhan Fungsional

Gambar 5.17 Implementasi antarmuka *modal form* Tambah Kebutuhan Fungsional



Gambar 5.18 Implementasi antarmuka halaman Daftar Kebutuhan Non-Fungsional



Gambar 5.19 Implementasi antarmuka *modal form* Tambah Kebutuhan Non-Fungsional

5.2.4.3 Implementasi Antarmuka Halaman Penentuan Prioritas Kebutuhan Non-Fungsional

Gambar 5.20 berikut ini merupakan hasil implementasi antarmuka halaman Prioritasi Kebutuhan Non-Fungsional.

5.2.4.4 Implementasi Antarmuka Halaman Prioritas Kebutuhan Non-Fungsional

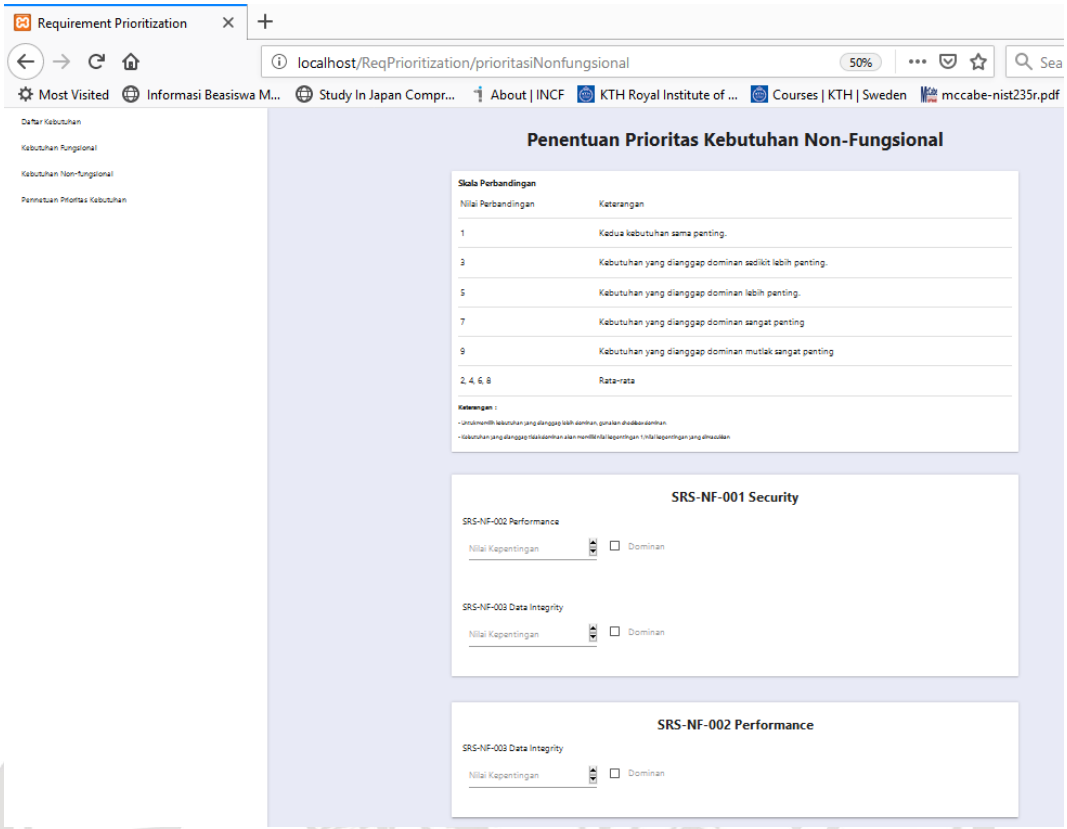
Gambar 5.21 berikut ini merupakan hasil implementasi antarmuka halaman Hasil Prioritasi Kebutuhan Non-Fungsional.

5.2.4.5 Implementasi Antarmuka Halaman Penentuan Prioritas Kebutuhan Fungsional

Gambar 5.22 berikut ini merupakan hasil implementasi antarmuka halaman Prioritasi Kebutuhan Fungsional.

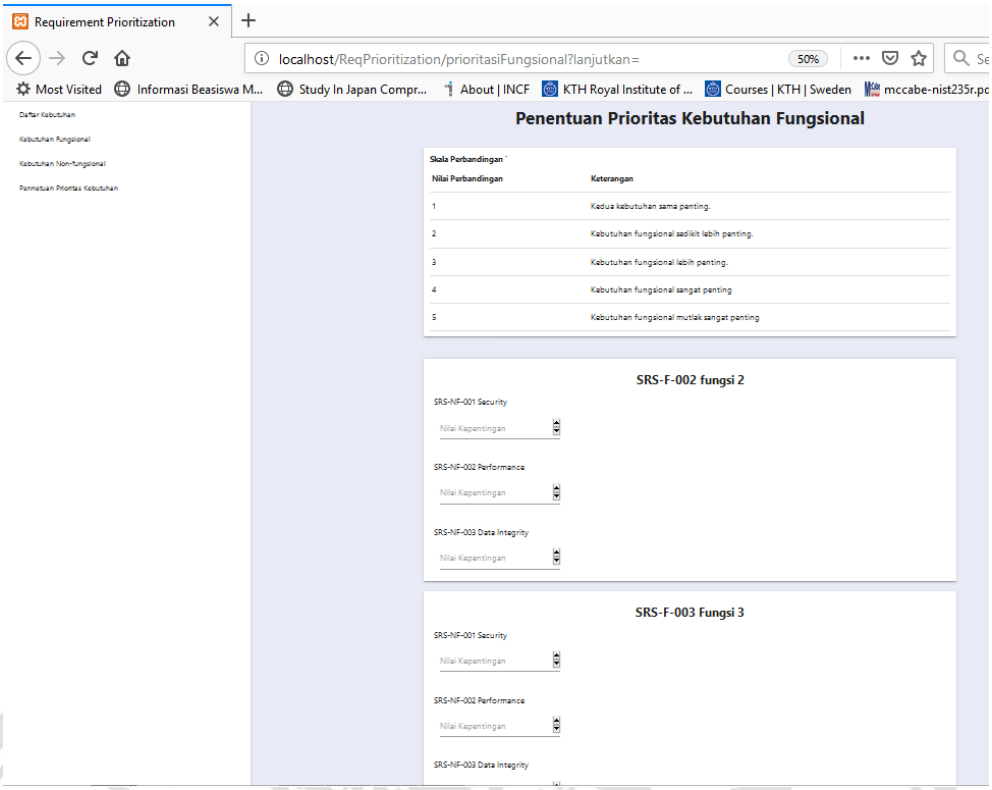
5.2.4.6 Implementasi Antarmuka Halaman Prioritas Kebutuhan Fungsional

Gambar 5.23 berikut ini merupakan hasil implementasi antarmuka halaman Hasil Prioritasi Kebutuhan Fungsional.



Gambar 5.20 Implementasi antarmuka halaman prioritasi kebutuhan non-fungsional

Gambar 5.21 Implementasi antarmuka halaman hasil prioritasi kebutuhan non-fungsional



Gambar 5.22 Implementasi antarmuka halaman prioritasi kebutuhan fungsional

Gambar 5.23 Implementasi antarmuka halaman hasil prioritasi kebutuhan fungsional

BAB 6 PENGUJIAN

6.1 Pengujian Unit

Pengujian unit dilakukan dengan teknik pengujian *white box* dan metode *basis path testing.* Pengujian dilakukan pada 3 sampel, yaitu pada Klas Kebutuhanfungsional dengan *method* tambahKebutuhan, Klas Kebutuhannonfungsional dengan *method* updatePrioritas, dan Klas DaftarKebutuhanNonfungsional dengan *method* getAllDesc.

6.1.1 Pengujian Unit Klas Kebutuhanfungsional *Method* tambahKebutuhan

Tabel 6.1 *Pseudocode Method* tambahKebutuhan

No	*Pseudocode*	
1	Start	
2	Membuat variabel associative array data (	
3	Inisialisasi index kode = kode	
4	Inisialisasi index deskripsi = deskripsi	
5	Inisialisasi index waktu = waktu	1
6	Inisialisasi index biaya = biaya	
7	)	
8	Query insert array $data ke tabel fungsional	
9	end	

Flow Graph

Gambar 6.1 *Flow graph method* tambahKebutuhan

***Cyclomatic Complexity* (V(G))**

V(G) = 1 *Regions*

V(G) = 0E – 1N + 2 = 1

V(G) = 0P + 1 = 1

Independent Path

Jalur 1 : 1

Kasus uji dan hasil uji dari *method* tambahKebutuhan terdapat pada Tabel 6.2.

Tabel 6.2 Kasus Uji dan Hasil Uji *Method* tambahKebutuhan

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	1. Klas *driver* unitTesting memanggil *method* tambahData	*Retun value* bernilai *true.*	*Retun value* bernilai *true.*	Valid

6.1.2 Pengujian Unit Klas DaftarKebutuhanNonfungsional *Method* getAllDesc

Tabel 6.3 *Pseudocode method* getAllDesc

No	*Pseudocode*
1	`Start`
2	`Query mengurutkan data kebutuhan non-fungsional berdasarkan nilai prioritas secara descending`
3	`Query mengambil seluruh data kebutuhan non-fungsional yang sudah berurutan`
4	`End`

Flow Graph

Gambar 6.2 *Flow graph method* getAllDesc

***Cyclmatic Complexity* (V(G))**

V(G) = 1 *Regions*

V(G) = 0E - 1N + 2 = 1

V(G) = 0P + 1 = 1

Independent Path

Jalur 1 : 1

Kasus uji dan hasil uji dari *method* getAllDesc terdapat pada Tabel 6.4.

Tabel 6.4 Kasus Uji dan Hasil Uji *Method* getAllDesc

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	1. Klas *driver* unitTesting memanggil *method* getAllDesc	*Retun value* bernilai *true.*	*Retun value* bernilai *true.*	Valid

6.1.3 Pengujian Unit Klas Kebutuhannonfungsional *Method* updatePrioritas

Tabel 6.5 *Pseudocode method* updatePrioritas

No	*Pseudocode*
1	`Start`
2	`inisialisasi variabel data = prioritas`
3	`Query find data kebutuhan non-fungsional deng  = id`
4	`Query update nilai prioritas kebutuhan non-fu onal dengan nilai pada variabel data`
5	`End`

1

Flow Graph

Gambar 6.3 *Flow graph method* updatePrioritas

***Cyclmatic Complexity* (V(G))**

V(G) = 1 *Region*

V(G) = 0E - 1N + 2 = 1

V(G) = 0P + 1 = 1

Independent Path

Jalur 1 : 1

Kasus uji dan hasil uji dari *method* updatePrioritas terdapat pada Tabel 6.6.

Tabel 6.6 Kasus Uji dan Hasil Uji *Method* updatePrioritas

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	1. Inisialisasi variabel data = 3.5 2. Inisialisasi variabel id = 1 3. Klas *driver* unitTesting memanggil *method* updatePrioritas	*Return value* bernilai *true.*	*Return value* bernilai *true.*	Valid

6.2 Pengujian Integrasi

Pengujian integrasi berada pada level *high-level design*, yaitu relasi antar klas dalam sistem. Teknik yang digunakan dalam pengujian integrasi ini adalah *white box testing* dengan metode *basis path testing*. Pengujian integrasi dilakukan pada *method* prosesPrioritasiNF pada klas AnalisController yang di dalamnya

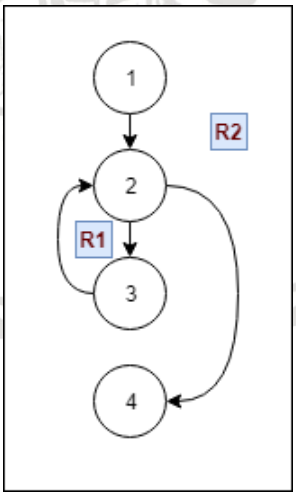
memanggil *method* updatePrioritas dari klas Kebutuhannonfungsional dan getAllDesc dari klas DaftarKebutuhanNonFungsional.

Tabel 6.7 *Pseudocode Method* prosesPrioritasNF

No	*Pseudocode*
1	Start
2	jumlah_nf = memanggil method getJumlahKebutuhan() dengan objek nonfungsional
3	id_nf = memanggil method getArrayID() dengan objek nonfungsional
4	input_perbandingan_nf = input post 'kepentingan'
5	input_dom = input post 'dom'
6	id_nf_utama = imput post 'id_utama'
7	id_nf_pembanding = imput post 'id_pembanding'
8	np = merge array input_perbandingan_nf
9	dom = merge array input_dom
10	Memanggil method simpanPerbadinganNF (id_nf_utama, id_nf_pembanding, np, dom)
11	prioritas_nf = hitungPrioritasNF(input_perbandingan_nf, input_dom, jumlah_nf)
12	for x=0 to jumlah_nf-1
13	memanggil method updatePrioritas(id_nf[x] oritas_nf[x]) dengan objek nonfungsional
14	endfor
15	nf = memanggil method getAllDesc dengan objek nonfungsion
16	Menampilkan halaman hasilPrioritasNF
17	End

Flow Graph

Flow graph untuk *method* prosesPrioritasNF terdapat pada Gambar 6.4.



Gambar 6.4 *Flow graph method* prosesPrioritasNF

***Cyclomatic Complexity* (V(G))**

V(G) = 2 *Regions*

V(G) = 4E – 4N + 2 = 2

V(G) = 1P + 1 = 2

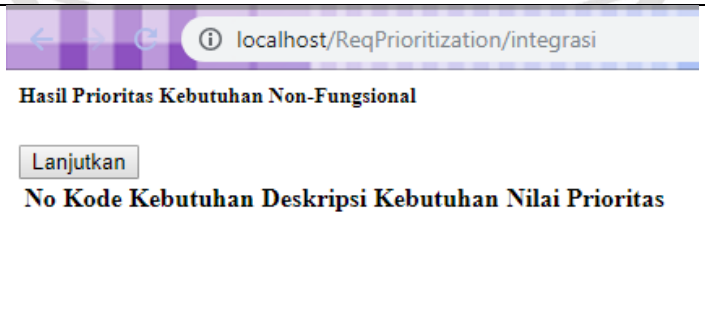
Independent Path

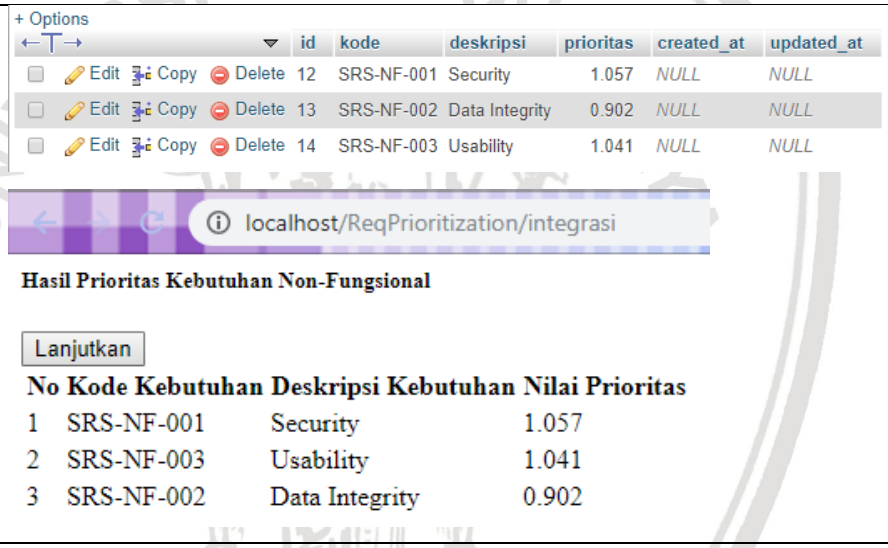
Jalur 1 : 1 – 2 – 4

Jalur 2 : 1 – 2 – 3 – 2 – 4

Kasus uji dan hasil pengujian integrasi *method* prosesPrioritasiNF terdapat pada Tabel6.8.

Tabel 6.8 Kasus Uji dan Hasil Uji Pengujian Integrasi *Method* prosesPrioritasiNF

No. Jalur	Prosedur Uji	*Expected Result*	*Result*	Status
1	1. Menghapus seluruh data kebutuhan non-fungsional pada database. 2. Inisialisasi variabel $input_perbandingan_nf = [[null, 3,4], [null, null, 2]] 3. Inisialisasi variabel $input_dom – [[null, “dom”, null], [null, null, “dom”]] 4. Set *return value method* hitungPrioritasNF dengan nilai [1.057, 0.902, 1.041] 5. Menjalankan *method* prosesPrioritasiNF	Tidak ada data kebutuhan non-fungsional yang ditampilkan oleh sistem.	Tidak ada data kebutuhan non-fungsional yang ditampilkan oleh sistem.	Valid
	Screenshot hasil pengujian			
				
2	1. Inisialisasi variabel $input_perbandingan_	Nilai prioritas kebutuhan	Nilai prioritas kebutuhan	Valid

<table>
<tr><td></td><td>nf = [[null, 3,4], [null, null, 2]]
2. Inisialisasi variabel $input_dom – [[null, “dom”, null], [null, null, “dom”]]
3. Set return value method hitungPrioritasNF dengan nilai [1.057, 0.902, 1.041]
4. Menjalankan method prosesPrioritasNF</td><td>non-fungsional pada database berhasil di update dan sistem menampilkan kebutuhan non-fungsional dari prioritas tertinggi ke prioritas terendah.</td><td>non-fungsional pada database berhasil di update dan sistem menampilkan kebutuhan non-fungsional dari prioritas tertinggi ke prioritas terendah.</td><td></td></tr>
<tr><td></td><td colspan="3">Screenshot hasil pengujian</td></tr>
<tr><td></td><td colspan="3">


</td></tr>
</table>

6.3 Pengujian Validasi

Pengujian validasi dilakukan dengan teknik pengujian *black box* dan metode *scenario-based testing*. Pengujian validasi dilakukan untuk memeriksa apakah sistem sudah memenuhi skenario yang sudah ada. Pengujian validasi memberikan hasil valid apabila sistem dapat memenuhi alur seperti pada skenario dan memberikan *output* yang sesuai berdasarkan data masukkan.

6.3.1 Pengujian Validasi Menambah Kebutuhan Fungsional

a. Kasus uji menambah kebutuhan fungsional berhasil

Tabel 6.9 berikut merupakan pengujian validasi untuk kasus uji menambah kebutuhan fungsional yang berhasil.

Tabel 6.9 Kasus Uji dan Hasil Uji Menambah Kebutuhan Fungsional Berhasil

Kode Kebutuhan	RPBN-F-01
Nama Kasus Uji	Menambah kebutuhan fungsional
Prosedur	1. Mengakses halaman daftar kebutuhan fungsional. 2. Mengklik tombol tambah kebutuhan fungsional. 3. Mengisi *form* tambah kebutuhan fungsional dengan data sebagai berikut : a. Deskripsi kebutuhan = Sistem menyediakan fungsi menambahkan data barang b. Estimasi waktu implementasi kebutuhan = 1 c. Estimasi biaya implementasi kebutuhan = 50.000 4. Mengklik tombol tambah.
Expected Result	Sistem mengarahkan ke halaman daftar kebutuhan fungsional dengan kebutuhan yang baru ditambahkan sudah masuk dalam tabel daftar kebutuhan.
Result	Sistem mengarahkan ke halaman daftar kebutuhan fungsional dengan kebutuhan yang baru ditambahkan sudah masuk dalam tabel daftar kebutuhan.
Status	Valid

b. Kasus uji menambah kebutuhan fungsional kondisi gagal

Tabel 6.10 berikut merupakan pengujian validasi untuk kasus uji menambah kebutuhan fungsional gagal karena kolom isian yang harus diisi dibiarkan kosong.

abel 6.10 Kasus Uji dan Hasil Uji Menambah Kebutuhan Fungsional Gagal

Kode Kebutuhan	RPBN-F-01
Nama Kasus Uji	Menambahkan kebutuhan fungsional alternatif 1
Prosedur	1. Mengakses halaman daftar kebutuhan fungsional. 2. Mengklik tombol tambah kebutuhan fungsional. 3. Mengklik tombol tambah.
Expected Result	Sistem menampilkan pesan untuk mengisi kolom isian pada *form*.
Result	Sistem menampilkan pesan untuk mengisi kolom isian pada *form*.
Status	Valid

c. Kasus uji menambah kebutuhan fungsional kondisi dibatalkan

Tabel 6.11 berikut merupakan pengujian validasi untuk kasus uji membatalkan penambahan kebutuhan fungsional.

Tabel 6.11 Kasus Uji dan Hasil Uji Menambah Kebutuhan Fungsional Batal

Kode Kebutuhan	RPBN-F-01
Nama Kasus Uji	Menambahkan kebutuhan fungsional alternatif 2
Prosedur	1. Mengakses halaman daftar kebutuhan fungsional. 2. Mengklik tombol tambah kebutuhan fungsional. 3. Mengisi *form* tambah kebutuhan fungsional dengan data sebagai berikut : a. Deskripsi kebutuhan = Sistem menyediakan fungsi penjumlahan b. Estimasi waktu implementasi kebutuhan = 1 c. Estimasi biaya implementasi kebutuhan = 50.000 4. Mengklik tombol batal.
Expected Result	Sistem mengarahkan ke halaman daftar kebutuhan fungsional tanpa menambahkan data baru di tabel kebutuhan fungsional.
Result	Sistem mengarahkan ke halaman daftar kebutuhan fungsional tanpa menambahkan data baru di tabel kebutuhan fungsional.
Status	Valid

6.3.2 Pengujian Validasi Menambah Kebutuhan Non-fungsional

a. Kasus uji menambah kebutuhan non-fungsional berhasil

Tabel 6.12 merupakan pengujian validasi untuk menambah kebutuhan non-fungsional yang berhasil.

Tabel 6.12 Kasus Uji dan Hasil Uji Menambah Kebutuhan Non-Fungsional

Kode Kebutuhan	RPBN-F-02
Nama Kasus Uji	Menambahkan kebutuhan non-fungsional
Prosedur	1. Mengakses halaman daftar kebutuhan non-fungsional. 2. Mengklik tombol tambah kebutuhan non-fungsional. 3. Mengisi kolom isian deskripsi kebutuhan dengan isian “Security”.

	4. Mengklik tombol tambah.
Expected Result	Sistem mengarahkan ke halaman daftar kebutuhan non-fungsional dengan kebutuhan yang baru ditambahkan sudah masuk dalam tabel daftar kebutuhan.
Result	Sistem mengarahkan ke halaman daftar kebutuhan non-fungsional dengan kebutuhan yang baru ditambahkan sudah masuk dalam tabel daftar kebutuhan.
Status	Valid

b. Kasus uji menambah kebutuhan non-fungsional gagal

Tabel 6.13 merupakan pengujian validasi menambah kebutuhan non-fungsional yang gagal karena ada kolom isian deskripsi kebutuhan dikosongkan.

Tabel 6.13 Kasus Uji dan Hasil Uji Menambah Kebutuhan Fungsional Gagal

Kode Kebutuhan	RPBN-F-02
Nama Kasus Uji	Menambahkan kebutuhan non-fungsional alternatif 1
Prosedur	1. Mengakses halaman daftar kebutuhan non-fungsional. 2. Mengklik tombol tambah kebutuhan non-fungsional. 3. Mengklik tombol tambah.
Expected Result	Sistem menampilkan pesan untuk mengisi kolom isian pada *form*.
Result	Sistem menampilkan pesan untuk mengisi kolom isian pada *form*.
Status	Valid

c. Kasus uji menambah kebutuhan non-fungsional kondisi dibatalkan

Tabel 6.14 merupakan pengujian validasi untuk membatalkan penambahan kebutuhan non-fungsional.

Tabel 6.14 Kasus Uji dan Hasil Uji Menambah Kebutuhan Fungsional Batal

Kode Kebutuhan	RPBN-F-02
Nama Kasus Uji	Menambahkan kebutuhan fungsional alternatif 2
Prosedur	1. Mengakses halaman daftar kebutuhan non-fungsional. 2. Mengklik tombol tambah kebutuhan non-fungsional. 3. Mengisi kolom isian deskripsi kebutuhan dengan isian "Sistem dapat diakses selama 24 jam".

	4. Mengklik tombol batal.
Expected Result	Sistem mengarahkan ke halaman daftar kebutuhan non-fungsional tanpa menambahkan data baru di tabel kebutuhan non-fungsional.
Result	Sistem mengarahkan ke halaman daftar kebutuhan non-fungsional tanpa menambahkan data baru di tabel kebutuhan non-fungsional.
Status	Valid

6.3.3 Pengujian Validasi Melihat Daftar Kebutuhan Fungsional

Pengujian validasi untuk melihat Daftar Kebutuhan Fungsional terdapat pada Tabel 6.15.

Tabel 6.15 Kasus Uji dan Hasil Uji Melihat Daftar Kebutuhan Fungsional

Kode Kebutuhan	RPBN-F-03
Nama Kasus Uji	Melihat daftar kebutuhan fungsional
Prosedur	1. Mengakses halaman daftar kebutuhan fungsional.
Expected Result	Sistem menampilkan daftar kebutuhan fungsional berurutan dari kebutuhan dengan nilai prioritas tertinggi hingga terendah.
Result	Sistem menampilkan daftar kebutuhan fungsional berurutan dari kebutuhan dengan nilai prioritas tertinggi hingga terendah.
Status	Valid

6.3.4 Pengujian Validasi Melihat Daftar Kebutuhan Non-fungsional

Pengujian validasi untuk melihat Daftar Kebutuhan Non-fungsional terdapat pada Tabel 6.16.

Tabel 6.16 Kasus Uji dan Hasil Uji Melihat Daftar Kebutuhan Non-fungsional

Kode Kebutuhan	RPBN-F-04
Nama Kasus Uji	Melihat daftar kebutuhan non-fungsional
Prosedur	1. Mengakses halaman daftar kebutuhan non-fungsional.
Expected Result	Sistem menampilkan daftar kebutuhan non-fungsional berurutan dari kebutuhan dengan nilai prioritas tertinggi hingga terendah.

Result	Sistem menampilkan daftar kebutuhan non-fungsional berurutan dari kebutuhan dengan nilai prioritas tertinggi hingga terendah.
Status	Valid

6.3.5 Pengujian Validasi Mengubah Kebutuhan Fungsional

a. Kasus uji mengubah kebutuhan fungsional berhasil

Tabel 6.17 merupakan pengujian validasi untuk mengubah kebutuhan fungsional yang berhasil.

Tabel 6.17 Kasus Uji dan Hasil Uji Mengubah Kebutuhan Fungsional Berhasil

Kode Kebutuhan	RPBN-F-05
Nama Kasus Uji	Mengubah kebutuhan fungsional
Prosedur	1. Mengakses halaman kebutuhan fungsional. 2. Mengklik tombol *edit* pada kebutuhan berikut : Kode kebutuhan SRS-F-001 Deskripsi kebutuhan : Sistem dapat menyediakan fungsi login Estimasi waktu implementasi kebutuhan : 1 Estimasi biaya implementasi kebutuhan : 0 3. Mengubah data kebutuhan ada form perubahan kebutuhan fungsional dengan data sebagai berikut : Kode kebutuhan SRS-F-001 Deskripsi kebutuhan : Sistem dapat menyediakan fungsi login Estimasi waktu implementasi kebutuhan : 1 Estimasi biaya implementasi kebutuhan : 50.000 4. Mengklik tombol simpan.
Expected Result	Perubahan disimpan dan ditampilkan di halaman daftar kebutuhan.
Result	Perubahan disimpan dan ditampilkan di halaman daftar kebutuhan.
Status	Valid

b. Kasus uji mengubah kebutuhan fungsional gagal

Tabel 6.18 merupakan pengujian validasi untuk mengubah kebutuhan fungsional yang gagal karena kolom isian yang harus diisi dikosongkan.

Tabel 6.18 Kasus Uji dan Hasil Uji Mengubah Kebutuhan Fungsional Gagal

Kode Kebutuhan	RPBN-F-05
Nama Kasus Uji	Mengubah kebutuhan fungsional alternatif 1
Prosedur	1. Mengakses halaman kebutuhan fungsional. 2. Mengklik tombol *edit* pada kebutuhan berikut : Kode kebutuhan SRS-F-002 Deskripsi kebutuhan : Sistem dapat menyediakan fungsi logout Estimasi waktu implementasi kebutuhan : 1 Estimasi biaya implementasi kebutuhan : 50.000 3. Mengubah data kebutuhan ada form perubahan kebutuhan fungsional dengan data sebagai berikut : Kode kebutuhan SRS-F-002 Deskripsi kebutuhan : (dikosongkan) Estimasi waktu implementasi kebutuhan : 1 Estimasi biaya implementasi kebutuhan : 50.000 4. Mengklik tombol simpan.
Expected Result	Sistem menampilkan pesan untuk mengisi kolom isian deskripsi kebutuhan.
Result	Sistem menampilkan pesan untuk mengisi kolom isian deskripsi kebutuhan.
Status	Valid

c. Kasus uji mengubah kebutuhan fungsional kondisi dibatalkan

Tabel 6.19 merupakan pengujian validasi untuk membatalkan perubahan kebutuhan fungsional.

Tabel 6.19 Kasus Uji dan Hasil Uji Mengubah Kebutuhan Fungsional Batal

Kode Kebutuhan	RPBN-F-05
Nama Kasus Uji	Mengubah kebutuhan fungsional alternatif 2
Prosedur	1. Mengakses halaman kebutuhan fungsional. 2. Mengklik tombol *edit* pada kebutuhan berikut : Kode kebutuhan SRS-F-001

	Deskripsi kebutuhan : Sistem dapat menyediakan fungsi login Estimasi waktu implementasi kebutuhan : 1 Estimasi biaya implementasi kebutuhan : 50.000 3. Mengklik tombol batal.
Expected Result	Sistem menampilkan daftar kebutuhan tanpa perubahan.
Result	Sistem menampilkan daftar kebutuhan tanpa perubahan.
Status	Valid

6.3.6 Pengujian Validasi Mengubah Kebutuhan Non-fungsional

a. Kasus uji mengubah kebutuhan non-fungsional berhasil

Tabel 6.20 merupakan pengujian validasi untuk mengubah kebutuhan non-fungsional yang berhasil.

Tabel 6.20 Kasus Uji dan Hasil Uji Mengubah Kebutuhan Non-fungsional Berhasil

Kode Kebutuhan	RPBN-F-06
Nama Kasus Uji	Mengubah kebutuhan non-fungsional
Prosedur	1. Mengakses halaman kebutuhan non-fungsional. 2. Mengklik tombol *edit* pada kebutuhan berikut: Kode kebutuhan SRS-NF-001 Deskripsi kebutuhan : Sistem dapat digunakan 24 jam 3. Mengubah kolom isian deskripsi kebutuhan dengan data berikut : Kode kebutuhan SRS-NF-001 Deskripsi kebutuhan : Availability 4. Mengklik tombol simpan.
Expected Result	Perubahan disiman dan ditampilkan di halaman daftar kebutuhan.
Result	Perubahan disiman dan ditampilkan di halaman daftar kebutuhan.
Status	Valid

b. Kasus uji mengubah kebutuhan non-fungsional gagal

Tabel 6.21 merupakan pengujian validasi untuk mengubah kebutuhan fungsional yang gagal karena kolom isian deskripsi kebutuhan dikosongkan.

Tabel 6.21 Kasus Uji dan Hasil Uji Mengubah Kebutuhan Non-fungsional Gagal

Kode Kebutuhan	RPBN-F-06
Nama Kasus Uji	Mengubah kebutuhan non-fungsional alternatif 1
Prosedur	a. Mengakses halaman kebutuhan non-fungsional. b. Mengklik tombol *edit* pada kebutuhan berikut : Kode kebutuhan SRS-NF-002 Deskripsi kebutuhan : Security c. Mengubah kolom isian deskripsi kebutuhan dengan data berikut : Kode kebutuhan SRS-NF-002 Deskripsi kebutuhan : (dikosongkan) d. Mengklik tombol simpan.
Expected Result	Sistem menampilkan pesan untuk mengisi kolom isian deskripsi kebutuhan.
Result	Sistem menampilkan pesan untuk mengisi kolom isian deskripsi kebutuhan.
Status	Valid

c. Kasus uji mengubah kebutuhan non-fungsional kondisi dibatalkan

Tabel 6.22 merupakan pengujian validasi untuk membatalkan perubahan kebutuhan fungsional.

Tabel 6.22 Kasus Uji dan Hasil Uji Mengubah Kebutuhan Non-fungsional Batal

Kode Kebutuhan	RPBN-F-06
Nama Kasus Uji	Mengubah kebutuhan fungsional alternatif 2
Prosedur	1. Mengakses halaman kebutuhan non-fungsional. 2. Mengklik tombol *edit* pada kebutuhan yang ingin diubah. 3. Mengklik tombol batal.
Expected Result	Sistem menampilkan daftar kebutuhan tanpa perubahan.
Result	Sistem menampilkan daftar kebutuhan tanpa perubahan.
Status	Valid

6.3.7 Pengujian Validasi Menghapus Kebutuhan Fungsional

a. Kasus uji menghapus kebutuhan fungsional berhasil

Tabel 6.23 merupakan pengujian validasi untuk menghapus kebutuhan fungsional yang berhasil.

Tabel 6.23 Kasus Uji dan Hasil Uji Menghapus Kebutuhan Fungsional Berhasil

Kode Kebutuhan	RPBN-F-07
Nama Kasus Uji	Menghapus kebutuhan fungsional.
Prosedur	1. Mengakses halaman daftar kebutuhan fungsional. 2. Mengklik tombol hapus pada kebutuhan dengan kode kebutuhan SRS-F-001. 3. Mengklik "Ya" pada pesan konfirmasi penghapusan.
Expected Result	Kebutuhan dengan kode kebutuhan SRS-F-001 berhasil di hapus dan tidak ditampilkan di daftar kebutuhan fungsional.
Result	Kebutuhan dengan kode kebutuhan SRS-F-001 berhasil di hapus dan tidak ditampilkan di daftar kebutuhan fungsional.
Status	Valid

b. Kasus uji menghapus kebutuhan fungsional kondisi dibatalkan

Tabel 6.24 merupakan pengujian validasi untuk membatalkan penghapusan kebutuhan fungsional.

Tabel 6.24 Kasus Uji dan Hasil Uji Menghapus Kebutuhan Fungsional Batal

Kode Kebutuhan	RPBN-F-07
Nama Kasus Uji	Menghapus kebutuhan fungsional alternatif 1.
Prosedur	1. Mengakses halaman daftar kebutuhan fungsional. 2. Mengklik tombol hapus pada kebutuhan dengan kode kebutuhan SRS-F-003. 3. Mengklik "tidak" pada pesan konfirmasi penghapusan.
Expected Result	Kebutuhan dengan kode kebutuhan SRS-F-003 tetap ditampilkan di daftar kebutuhan fungsional.
Result	Kebutuhan dengan kode kebutuhan SRS-F-003 tetap ditampilkan di daftar kebutuhan fungsional.
Status	Valid

6.3.8 Pengujian Validasi Menghapus Kebutuhan Non-fungsional

a. Kasus uji menghapus kebutuhan non-fungsional berhasil

Tabel 6.25 merupakan pengujian validasi untuk menghapus kebutuhan non-fungsional yang berhasil.

Tabel 6.25 Kasus Uji dan Hasil Uji Menghapus Kebutuhan Non-fungsional Berhasil

Kode Kebutuhan	RPBN-F-08
Nama Kasus Uji	Menghapus kebutuhan non-fungsional.
Prosedur	1. Mengakses halaman daftar kebutuhan non-fungsional. 2. Mengklik tombol hapus pada kebutuhan dengan kode kebutuhan SRS-NF-003 3. Mengklik "ya" pada pesan konfirmasi penghapusan.
Expected Result	Kebutuhan dengan kode kebutuhan SRS-NF-003 berhasil di hapus dan tidak ditampilkan di daftar kebutuhan non-fungsional.
Result	Kebutuhan dnegan kode kebutuhan SRS-NF-003 berhasil di hapus dan tidak ditampilkan di daftar kebutuhan non-fungsional.
Status	Valid

b. Kasus uji menghapus kebutuhan non-fungsional kondisi dibatalkan

Tabel 6.26 merupakan pengujian validasi untuk membatalkan penghapusan kebutuhan fungsional.

Tabel 6.26 Kasus Uji dan Hasil Uji Menghapus Kebutuhan Non-fungsional Batal

Kode Kebutuhan	RPBN-F-08
Nama Kasus Uji	Menghapus kebutuhan non-fungsional alternatif 1.
Prosedur	1. Mengakses halaman daftar kebutuhan non-fungsional. 2. Mengklik tombol hapus pada kebutuhan dengan kode kebutuhan SRS-NF-001 3. Mengklik "tidak" pada pesan konfirmasi penghapusan.
Expected Result	Kebutuhan dengan kode kebutuhan SRS-NF-001 tetap ditampilkan di daftar kebutuhan non-fungsional.
Result	Kebutuhan dengan kode kebutuhan SRS-NF-001 tetap ditampilkan di daftar kebutuhan non-fungsional.
Status	Valid

6.3.9 Pengujian Validasi Menghitung Prioritas Kebutuhan

a. Kasus uji menghitung prioritas kebutuhan berhasil

Tabel 6.27 merupakan pengujian validasi untuk menghitung prioritas kebutuhan yang berhasil.

Tabel 6.27 Kasus Uji dan Hasil Uji Menghitung Prioritas Kebutuhan Berhasil

Kode Kebutuhan	RPBN-F-09
Nama Kasus Uji	Menghitung prioritas kebutuhan
Prosedur	1. Mangakses halaman prioritasi kebutuhan. 2. Mengisi kolom isian kepentingan dengan nilai 3, 5, 4 dan memilih cekbox dominan pada perbandingan antara kebutuhan SRS-NF-001 dengan kebutuhan SRS-NF-003 juga pada perbandingan kebutuhan SRS-NF-002 dengan kebutuhan SRS-NF-003. 3. Mengklik tombol “Lanjutkan” 4. Mengklik tombol “Lanjutkan” pada halaman hasil prioritasi kebutuahan non-fungsional. 5. Mengisi kolom kepentingan pada perbandingan kebutuhan fungsional dengan nilai 3, 1, 3 untuk perbandingan kebutuhan fungsional SRS-F-001, dan nilai 5, 3, 1 untuk perbandingan kebutuhan fungsional SRS-F-002. 6. Megklik tombol “Selesai”.
Expected Result	3.1 Sistem menampilkan hasil prioritasi kebutuhan non-fungsional, dengan nilai prioritas kebutuhan non-fungsional sebagai berikut : SRS-NF-001 : 0.671 SRS-NF-002 : 0.35 SRS-NF-003 : 1.979 6.1 Sistem menampilkan hasil prioritasi kebutuhan fungsional dengan nilai prioritas kebutuhan fungsional sebagai berikut : SRS-F-001 : 6.321 SRS-F-002 : 6.384
Result	3.1 Sistem menampilkan hasil prioritasi kebutuhan non-fungsional, dengan nilai prioritas kebutuhan non-fungsional sebagai berikut : SRS-NF-001 : 0.671 SRS-NF-002 : 0.35 SRS-NF-003 : 1.979

	6.1 Sistem menampilkan hasil prioritasi kebutuhan fungsional dengan nilai prioritas kebutuhan fungsional sebagai berikut : SRS-F-001 : 6.321 SRS-F-002 : 6.384
Status	Valid

b. Kasus uji menghitung prioritas kebutuhan kondisi belum ada data kebutuhan non-fungsional dalam *database*

Tabel 6.28 merupakan pengujian validasi untuk menghitung prioritas kebutuhan saat kondisi belum terdapat kebutuhan non-fungsional dalam database.

Tabel 6.28 Kasus Uji dan Hasil Uji Menghitung Prioritas Kebutuhan Kondisi Belum Ada Data Kebutuhan Non-fungsional

Kode Kebutuhan	RPBN-F-09
Nama Kasus Uji	Menghitung prioritas kebutuhan alternatif 1
Prosedur	.1 Mengakses halaman prioritasi kebutuhan
Expected Result	Sistem menampilkan pesan “Tidak ada kebutuhan non-fungsional. Masukkan kebutuhan non-fungsional terlebih dahulu.” dan menyediakan tombol “Kebutuhan Non-fungsional” untuk mengakses halaman kebutuhan non-fungsional.
Result	Sistem menampilkan pesan “Tidak ada kebutuhan non-fungsional. Masukkan kebutuhan non-fungsional terlebih dahulu.” dan menyediakan tombol “Kebutuhan Non-fungsional” untuk mengakses halaman kebutuhan non-fungsional.
Status	Valid

c. Kasus uji menghitung prioritas kebutuhan kondisi input nilai kepentingan non-fungsional lebih kecil dari 1

Tabel 6.29 merupakan pengujian validasi untuk menghitung prioritas kebutuhan dengan kondisi input yang lebih kecil dari 1.

Tabel 6.29 Kasus Uji dan Hasil Uji Menghitung Prioritas Kebutuhan Kondisi Nilai Kepentingan Lebih Kecil Dari 1

Kode Kebutuhan	RPBN-F-09
Nama Kasus Uji	Menghitung prioritas kebutuhan alternatif 2

Prosedur	1. Mangakses halaman prioritasi kebutuhan. 2. Mengisi kolom isian kepentingan dengan nilai 0, -5, -4. 3. Mengklik tombol “Lanjutkan”
Expected Result	Sistem menampilkan pesan “*Value must be greater than or equal to 1*”
Result	Sistem menampilkan pesan “*Value must be greater than or equal to 1*”
Status	Valid

d. Kasus uji menghitung prioritas kebutuhan kondisi input nilai kepentingan non-fungsional lebih besar dari 9

Tabel 6.30 merupakan pengujian validasi untuk menghitung prioritas kebutuhan dengan kondisi nilai input lebih besar dari 9.

Tabel 6.30 Kasus Uji dan Hasil Uji Menghitung Prioritas Kebutuhan Kondisi Nilai Kepentingan Lebih Besar dari 9

Kode Kebutuhan	RPBN-F-09
Nama Kasus Uji	Menghitung prioritas kebutuhan alternatif 3
Prosedur	1. Mangakses halaman prioritasi kebutuhan. 2. Mengisi kolom isian kepentingan dengan nilai 10, 15, 14. 3. Mengklik tombol “Lanjutkan”
Expected Result	Sistem menampilkan pesan “*Value must be less than or equal to 1*”
Result	Sistem menampilkan pesan “*Value must be greater or equal to 1*”
Status	Valid

e. Kasus uji menghitung prioritas kebutuhan kondisi input nilai kepentingan non-fungsional dikosongkan

Tabel 6.31 merupakan pengujian validasi untuk menghitung prioritas kebutuhan dengan kondisi kolom isian kepentingan dibiarkan kosong.

Tabel 6.31 Kasus Uji dan Hasil Uji Menghitung Prioritas Kebutuhan Kondisi Nilai Kepentingan DIkosongkan

Kode Kebutuhan	RPBN-F-09
Nama Kasus Uji	Menghitung prioritas kebutuhan alternatif 4
Prosedur	1. Mangakses halaman prioritasi kebutuhan.

	2. Mengosongkan kolom kepentingan terakhir pada *form*. 3. Mengklik tombol “Lanjutkan”
Expected Result	Sistem menampilkan pesan “*Please fill out thid field*”.
Result	Sistem menampilkan pesan “*Please fill out thid field*”.
Status	Valid

f. Kasus uji menghitung prioritas kebutuhan kondisi belum ada data kebutuhan fungsional dalam *database*

Tabel 6.32 merupakan pengujian validasi untuk menghitung prioritas kebutuhan dengan kondisi tidak ada data kebutuhan fungsional dalam database.

Tabel 6.32 Kasus Uji dan Hasil Uji Menghitung Prioritas Kebutuhan Kondisi Belum Ada Data Kebutuhan Fungsional

Kode Kebutuhan	RPBN-F-09
Nama Kasus Uji	Menghitung prioritas kebutuhan alternatif 5
Prosedur	1. Mengakses halaman prioritasi kebutuhan. 2. Mengisi kolom isian kepentingan dengan nilai 3, 5, 4 dan memilih cekbox dominan pada perbandingan antara kebutuhan SRS-NF-001 dengan kebutuhan SRS-NF-003 juga pada perbandingan kebutuhan SRS-NF-002 dengan kebutuhan SRS-NF-003. 3. Mengklik tombol “Lanjutkan” 4. Mengklik tombol “Lanjutkan” pada halaman hasil prioritasi kebutuahan non-fungsional.
Expected Result	Sistem menampilkan pesan “Tidak ada kebutuhan fungsional. Masukkan kebutuhan fungsional terlebih dahulu.” dan menyediakan tombol “Kebutuhan Fungsional” untuk mengakses halaman kebutuhan fungsional.
Result	Sistem menampilkan pesan “Tidak ada kebutuhan fungsional. Masukkan kebutuhan fungsional terlebih dahulu.” dan menyediakan tombol “Kebutuhan Fungsional” untuk mengakses halaman kebutuhan fungsional.
Status	Valid

g. Kasus uji menghitung prioritas kebutuhan kondisi input nilai kepentingan fungsional lebih kecil dari 1

Tabel 6.33 merupakan pengujian validasi untuk menghitung prioritas kebutuhan dengan kondisi input nilai perbandngan lebih kecil dari 1.

Tabel 6.33 Kasus Uji dan Hasil Uji Menghitung Prioritas Kebutuhan Kondisi Nilai Kepentingan Fungsional Lebih Kecil dari 1

Kode Kebutuhan	RPBN-F-09
Nama Kasus Uji	Menghitung prioritas kebutuhan alternatif 6
Prosedur	1. Mengakses halaman prioritasi kebutuhan. 2. Mengisi kolom isian kepentingan dengan nilai 3, 5, 4 dan memilih cekbox dominan pada perbandingan antara kebutuhan SRS-NF-001 dengan kebutuhan SRS-NF-003 juga pada perbandingan kebutuhan SRS-NF-002 dengan kebutuhan SRS-NF-003. 3. Mengklik tombol "Lanjutkan" 4. Mengklik tombol "Lanjutkan" pada halaman hasil prioritasi kebutuahan non-fungsional. 5. Mengisi kolom kepentingan pada perbandingan kebutuhan fungsional dengan nilai 0, -1, -3 untuk perbandingan kebutuhan fungsional SRS-F-001, dan nilai -5, -3, -1 untuk perbandingan kebutuhan fungsional SRS-F-002. 6. Megklik tombol "Selesai".
Expected Result	Sistem menampilkan pesan "*Value must be greater than or equal to 1*"
Result	Sistem menampilkan pesan "*Value must be greater than or equal to 1*"
Status	Valid

h. Kasus uji menghitung prioritas kebutuhan kondisi input nilai kepentingan non-fungsional lebih besar dari 5

Tabel 6.34 merupakan pengujian validasi untuk menghitung prioritas kebutuhan dengan kondisi input nilai perbandngan lebih besar dari 5.

Tabel 6.34 Kasus Uji dan Hasil Uji Menghitung Prioritas Kebutuhan Kondisi Nilai Kepentingan Fungsional Lebih Besar dari 5

Kode Kebutuhan	RPBN-F-09
Nama Kasus Uji	Menghitung prioritas kebutuhan alternatif 7
Prosedur	1. Mengakses halaman prioritasi kebutuhan. 2. Mengisi kolom isian kepentingan dengan nilai 3, 5, 4 dan memilih cekbox dominan pada perbandingan antara kebutuhan SRS-NF-001 dengan kebutuhan SRS-NF-003

	juga pada perbandingan kebutuhan SRS-NF-002 dengan kebutuhan SRS-NF-003. 3. Mengklik tombol “Lanjutkan” 4. Mengklik tombol “Lanjutkan” pada halaman hasil prioritasi kebutuahan non-fungsional. 5. Mengisi kolom kepentingan pada perbandingan kebutuhan fungsional dengan nilai 6, 2, 3 untuk perbandingan kebutuhan fungsional SRS-F-001, dan nilai 5, 3, 1 untuk perbandingan kebutuhan fungsional SRS-F-002. 6. Megklik tombol “Selesai”.
Expected Result	Sistem menampilkan pesan “*Value must be less than or equal to 5*” pada kolom yang diisi dengan nilai 6.
Result	Sistem menampilkan pesan “*Value must be less than or equal to 5*” pada kolom yang diisi dengan nilai 6.
Status	Valid

i. Kasus uji menghitung prioritas kebutuhan kondisi input nilai kepentingan fungsional dikosongkan

Tabel 6.35 merupakan pengujian validasi untuk menghitung prioritas kebutuhan dengan kondiis kolom isian nilai perbandingan unutuk kebutuhan dengan kode SRS-F-002 dikosongkan.

Tabel 6.35 Kasus Uji dan Hasil Uji Menghitung Prioritas Kebutuhan Kondisi Nilai Kepentingan Fungsional dikosongkan

Kode Kebutuhan	RPBN-F-09
Nama Kasus Uji	Menghitung prioritas kebutuhan alternatif 8
Prosedur	1. Mengakses halaman prioritasi kebutuhan. 2. Mengisi kolom isian kepentingan dengan nilai 3, 5, 4 dan memilih cekbox dominan pada perbandingan antara kebutuhan SRS-NF-001 dengan kebutuhan SRS-NF-003 juga pada perbandingan kebutuhan SRS-NF-002 dengan kebutuhan SRS-NF-003. 3. Mengklik tombol “Lanjutkan” 4. Mengklik tombol “Lanjutkan” pada halaman hasil prioritasi kebutuahan non-fungsional. 5. Mengisi kolom kepentingan pada perbandingan kebutuhan fungsional dengan nilai 6, 2, 3 untuk perbandingan kebutuhan fungsional SRS-F-001, dan

	mengosongkan kolom kepentingan untuk perbandingan kebutuhan fungsional SRS-F-002. 6. Megklik tombol “Selesai”.
Expected Result	Sistem menampilkan pesan “*Please fill out thid field”* pada kolom isian kepentingan yang kosong.
Result	Sistem menampilkan pesan “*Please fill out thid field”* pada kolom isian kepentingan yang kosong.
Status	Valid

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan yang didapat dari penelitian ini adalah sebagai berikut:

1. Rekayasa kebutuhan menghasilkan satu aktor yang berperan dalam sistem, yaitu analis, serta 9 kebutuhan fungsional yang digambarkan dalam bentuk *use case diagram*. Kebutuhan utama sistem adalah menghitung prioritas kebutuhan perangkat lunak.
2. Perancangan sistem menghasilkan tiga sampel pemodelan *sequence diagram*, pemodelan *class diagram* dengan 11 klas, perancangan komponen berupa *pseudocode,* perancangan data berupa CDM dan PDM, dan perancangan antarmuka berupa *wireframe*. Klas utama sistem adalah klas CI_Controller dan CI_Model. Hasil perancangan diimplementasikan ke dalam bentuk kode program, basis data, serta antarmuka sistem. Implementasi dilakukan dengan menggunakan *framework* CodeIgniter dan MaterializeCSS. *Database* yang digunakan adalah MySql.
3. Pengujian yang dilakukan adalah pengujian unit, pengujian integrasi, dan pengujian validasi. Hasil dari setiap pengujian menunjukkan hasil valid pada setiap kasus uji, sehingga bisa dikatakan sistem sudah memenuhi semua kebutuhan yang diperlukan.

7.2 Saran

Saran yang dapat diberikan untuk penelitian selanjutnya antara lain:

1. Aplikasi penentuan prioritas kebutuhan fungsional perangkat lunak berdasarkan kebutuhan non-fungsional ini dapat dikembangkan lebih jauh untuk keperluan manajemen kebutuhan perangkat lunak atau pun manajemen proyek perangkat lunak.

DAFTAR RFERENSI

Aasem, M., Ramzan, M. & Jaffar, A., 2010. *Analysis and Optimization of Software Requirements Prioritization Techniques.* Karachi, IEEE.

Andre, 2014. *Duniailkom.* [Online] Available at: https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-polimorfisme-dalam-pemrograman-objek-php/ [Accessed 25 February 2018].

Capretz, L. F., 2003. *Research Gate.* [Online] Available at: https://www.researchgate.net/publication/220630952_A_brief_history_of_the_object-oriented_approach [Accessed 25 February 2018].

Elysium Academy Private Limited, 2017. *LinkedIn.* [Online] Available at: https://www.linkedin.com/pulse/what-software-development-life-cycle-sdlc-phases-private-limited [Accessed 13 February 2018].

Fowler, M., 2003. *UML Distilled.* 3rd ed. s.l.:Addison-Wesley Professional.

Garg, U. & Singhal, A., 2017. *Software Requirement Prioritization based on Non-Functional Requirements.* Noida, IEEE.

Herrmann, A. & Daneva, M., 2008. *Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research.* Catalunya, IEEE.

Herrmann, A. & Daneva, M., 2008. *Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research.* Cartalunya, IEEE.

Jawale, B. B., Patnaik, G. K. & Bhole, A. T., 2017. *Requirement Prioritization using Adaptive Fuzzy Hierarchical Cumulative Voting.* Hyderabad, IEEE.

Karlsson, J. & Ryan, K., 1997. A cost-value approach for prioritizing requirements. *IEEE Software,* 14(5), pp. 67-74.

Latifah, S., 2005. *Perpustakaan Universitas Sumatera Utara.* [Online] Available at: http://library.usu.ac.id/download/fp/hutan-siti11.pdf [Accessed 13 February 2018].

Liaqat, R. M., Azam, F., Ahmed, M. A. & Mehboob, B., 2016. *A Majority Voting Goal Based Technique for Requirement Prioritization.* Colchester, IEEE.

Longani, P., Singjai, A. & Sitthithanasakul, S., 2017. *Software requirement workshops and outcomes: Experienced from software requirement analysis course for undergraduate.* London, IEEE.

MDN, 2017. *MDN Web Docs.* [Online] Available at: https://developer.mozilla.org/en-

US/docs/Web/JavaScript/About_JavaScript
[Accessed 13 October 2017].

Mocenni, C., 2007. *Department of Information Engineering and Mathematics - University of Siena.* [Online] Available at: http://www.dii.unisi.it/~mocenni/Note_AHP.pdf [Accessed 4 February 2018].

N., 2017. *Stacktips.* [Online] Available at: http://stacktips.com/tutorials/laravel/intro-to-laravel-php-framework-and-features [Accessed 14 February 2018].

Pandey, D., Suman, U. & Ramani, A., 2010. *An Effective Requirement Engineering Process Model for Software Development and Requirements Management.* Kottayam, IEEE.

Perini, A., Susi, A. & Avesani, P., 2013. A Machine Learning Approach to Software Requirement Prioritization. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,* 39(4), pp. 445-461.

Pressman, R. S., 2001. *Software Engineering A Practitioner's Approach.* 5th ed. New York: Thomas Casson.

Pressman, R. S. & Maxim, B. R., 2013. *Software Engineering a Practitioner's Approach.* 8 ed. New York: McGraw-Hill Education.

Ramzan, M., Jaffar, M. A. & Shahid, A. A., 2011. Value Based Intelligent Requirement Prioritization (VIRP): Expert Driven Fuzzy Logic Based Prioritization Technique. *International Journal of Innovative Computing, Information and Control,* 7(3), pp. 1017-1038.

Saaty, T. L., 1994. How to Make a Decision: The Analytic Hierarchy Process. *Interfaces,* 24(6), pp. 19-43.

Saaty, T. L., 2008. Decision Making With The Analytic Hierarchy Process. *Int. J. Services Sciences,* 1(1), pp. 83-98.

Sadiq, M. et al., 2010. *More on Elicitation of Software Requirements and Prioritization Using AHP.* Bangalore, IEEE.

Satzinger, J. S., Jackson, R. B. & Burd, S. D., 2012. *Systems Analysis and Design in a Changing World.* 6th ed. Boston: Joe Sabatino.

Shannon, R., 2012. *HTML Source.* [Online] Available at: http://www.yourhtmlsource.com/starthere/whatishtml.html [Accessed 26 February 2018].

Siahaan, D., 2012. *Analisa Kebutuhan dalam Rekayasa Perangkat Lunak.* 1st ed. Yogyakarta: Andi.

Sommerville, I., 2011. *Software Engineering.* 9th ed. Boston: Addison-Wesley.

Statista, 2018. *Global digital population as of January 2018 (in millions).* [Online] Available at: https://www.statista.com/statistics/617136/digital-population-worldwide/ [Accessed 1 February 2018].

The PHP Group, n.d. *PHP.* [Online] Available at: http://php.net/manual/en/intro-whatis.php [Accessed 25 February 2018].

uml-diagrams.org, 2017. *UML Class Diagrams Reference.* [Online] Available at: UML Class Diagrams Reference [Accessed 25 February 2018].

uml-diagrams.org, 2017. *UML Sequence Diagrams.* [Online] Available at: https://www.uml-diagrams.org/sequence-diagrams.html [Accessed 25 February 2018].

uml-diagrams.org, 2017. *Use Case Diagrams References.* [Online] Available at: https://www.uml-diagrams.org/use-case-reference.html [Accessed 25 February 2018].

W3C, n.d. *W3C.* [Online] Available at: https://www.w3.org/standards/webdesign/htmlcss [Accessed 13 October 2017].

W3Schools, 2017. *w3schools.com.* [Online] Available at: https://www.w3schools.com/css/css_howto.asp [Accessed 13 October 2017].

W3Schools, n.d. *W3Schools.com.* [Online] Available at: https://www.w3schools.com/html/html5_intro.asp [Accessed 26 February 2018].

Zhang, S. & Liu, Z., 2017. *Research on the construction and robustness testing of SaaS cloud computing data center based on the MVC design pattern.* Coimbatore, IEEE.